LECTURE NOTE

ON

WEB APPLICATION DEVELOPMENT

COURSE CODE: CSC3301

Compiled By

ADAM SALEH ADAM

Department of Computer Science,
Yobe State University,
Damaturu – Nigeria

2023

CSC3301 - Web Application Development

Course Content:

Introduction to framework-based web development using a contemporary language like PHP and ASP.NET. Principles of web pages (dynamic and static) and website design. The tool used in web development. Client-side and server-side languages. Creation of interactive, dynamic websites using a common web architecture and object-based database access. Design, implementation, and testing of web-based applications including related software, databases, interfaces, and digital media. Standard object models, and the use of server-side programmes for database and file access; testing, software quality assurance; and the process of publishing Web sites. Hands-on PHP and Python programme using open-source software (Apache, PHP, Python, JavaScript, and MySQL). Programming for web development includes control structures, objects, functions, and the use of composite data types. Deploying dynamic content using JavaScript. Designing and developing dynamic web pages and creating, validating, transforming, and formatting data using PHP.

Course Objectives:

At the end of the course, the students should be able to:

1. Design and implement simple client-side and server-side web applications;
2. Demonstrate hands-on skills in PHP or Python programming uses open-source software;
3. Compare and contrast web programming with general-purpose programming; and
4. Develop a fully functioning website and deploy it on a web server.

Course Structure:

UNIT - I

Web Basics and Overview: The Internet, World Wide Web, Web Browsers, URL , HTTP and Web Application Development.

UNIT – II

Frontend (Client-side) Development: Introduction to frontend web development using frontend web technologies and Frameworks.

UNIT – III

Backend (Server-side) Development: Introduction to backend web development using backend web technologies and Frameworks.

UNIT – IV

Implementation: Designing and developing a simple dynamic web application using PHP and deploying it on a web server.

TABLE OF CONTENTS

UNIT – I

Web Basics and Overview: The Internet, World Wide Web, Web Browsers, URL , HTTP and Web Application Development.

(a) The Internet

It is a global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols. The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries a vast range of information resources and services.

(b) Brief History of Internet

The history of the Internet can be traced back to the 1950s and 1960s, when the US military and government agencies began funding research into packet-switching and other technologies that would eventually form the foundation of the Internet.

In the late 1960s, the **Advanced Research Projects Agency Network (ARPANET)** was created by the US Department of Defense as an experimental wide-area network that used packet-switching to transmit data. This was the first operational packet-switching network, and it is considered to be the precursor to the modern Internet.

In the 1970s, the **Internet Protocol (IP)** was developed, which allowed different networks to communicate with each other. This was followed by the development of the **Transmission Control Protocol (TCP)**, which ensured that data packets were delivered reliably. Together, these protocols formed the foundation of the Internet Protocol Suite (TCP/IP), which is still in use today.

In the 1980s, other networks such as the National Physical Laboratory Network (NPL) in the UK, Cyclades in France, and the Merit Network in the US, adopted the TCP/IP protocols and interconnected to form the global Internet we know today.

In 1989, the World Wide Web (WWW) was created as a way to access and share information over the Internet using hypertext links. The first website went online in 1991, and the web's popularity exploded in the 1990s, with the introduction of the first web browsers.

The Internet has since become a global network connecting billions of people and devices, and has had a profound impact on many aspects of our lives, including communication, commerce, and entertainment.

(c) World Wide Web (WWW)

The World Wide Web, often referred to simply as **"the web,"** was created in 1989 by **Tim Berners-Lee**, a computer scientist while working at CERN. It was designed as a way to easily share and access information over the internet. The first web browser, called **WorldWideWeb** (**Nexus)**, was created by Berners-Lee in 1990. These early browsers were basic and text-based, but they laid the foundation for the more sophisticated web browsers that we use today.

In 1993, Marc Andreessen and Eric Bina at the National Center for Supercomputing Applications (NCSA) released **Mosaic**, one of the first graphical web browsers. Mosaic was easy to use and supported images, making the web more accessible to a wider audience. Mosaic's success led to the development of other popular browsers such as **Netscape Navigator** and **Internet Explorer**.

In the late 1990s and early 2000s, the web browser market became increasingly competitive, with new browsers such as **Opera** and **Safari** being released. In 2004, Mozilla released **Firefox**, which quickly gained popularity due to its speed, security, and customization options. Google released **Chrome** in 2008, which quickly became the most widely used web browser due to its speed and simplicity.

Today, the most widely used web browsers are Chrome, Safari, Firefox, and Microsoft Edge. These browsers continue to evolve and improve, with new features such as increased security, support for new web technologies, and improved performance.

(d) Web Browsers

They are software programs used to access the WWW. It is known as a browser because it "browses" the WWW and requests the hypertext documents. Browsers can be graphical, allowing the user to see and hear the graphics and audio files. Text-only browsers (i.e., those with no sound or graphics capability) are also available. All of these programs understand http and other Internet protocols such as FTP, gopher, mail, and news, making the WWW a kind of "one stop shopping" for Internet users.

Below is a table with the list of web browsers with their versions.

Year	Web Browsers and Versions
1991	World Wide Web (Nexus)
1992	Viola WWW, Erwise, MidasWWW, MacWWW (Samba)
1993	Mosaic, Cello,[2] Lynx 2.0, Arena, AMosaic 1.0
1994	IBM WebExplorer, Netscape Navigator, SlipKnot 1.0, MacWeb, IBrowse, Agora (Argo), Minuet
1995	Internet Explorer 1, Internet Explorer 2, Netscape Navigator 2.0, OmniWeb, UdiWWW, Grail
1996	Arachne 1.0, Internet Explorer 3.0, Netscape Navigator 3.0, Opera 2.0, PowerBrowser 1.5,[4] Cyberdog, Amaya 0.9,[5] AWeb, Voyager
1997	Internet Explorer 4.0, Netscape Navigator 4.0, Netscape Communicator 4.0, Opera 3.0,[6] Amaya 1.0[5]
1998	iCab, Mozilla
1999	Amaya 2.0,[5] Mozilla M3, Internet Explorer 5.0
2000	Konqueror, Netscape 6, Opera 4,[7] Opera 5,[8] K-Meleon 0.2, Amaya 3.0,[5] Amaya 4.0[5]
2001	Internet Explorer 6, Galeon 1.0, Opera 6,[9] Amaya 5.0[5]
2002	Netscape 7, Mozilla 1.0, Phoenix 0.1, Links 2.0, Amaya 6.0,[5] Amaya 7.0[5]
2003	Opera 7,[10] Apple Safari 1.0, Epiphany 1.0, Amaya 8.0[5]
2004	Firefox 1.0, Netscape Browser, OmniWeb 5.0
2005	Opera 8,[11] Apple Safari 2.0, Netscape Browser 8.0, Epiphany 1.8, Amaya 9.0,[5] AOL Explorer 1.0, Maxthon 1.0,Shiira 1.0
2006	Mozilla Firefox 2.0, Internet Explorer 7, Opera 9,[12], SeaMonkey 1.0, K-Meleon 1.0, Galeon 2.0, Camino 1.0, Avant11, iCab 3
2007	Apple Safari 3.0, Maxthon 2.0, Netscape Navigator 9, NetSurf 1.0, Flock 1.0, Conkeror
2008	Google Chrome 1, Mozilla Firefox 3, Opera 9.5,[13], Apple Safari 3.1, Konqueror 4, Amaya 10.0,[5] Flock 2, Amaya 11.0[5]
2009	Google Chrome 2 – 3, Mozilla Firefox 3.5, Internet Explorer 8, Opera 10,[14], Apple Safari 4, SeaMonkey 2, Camino 2,surf, Pale Moon 3.0[15]
2010	Google Chrome 4 – 8, Mozilla Firefox 3.6, Opera 10.50,[16], Opera 11, Apple Safari 5, K Meleon 1.5.4,
2011	Google Chrome 9 – 16, Mozilla Firefox 4-9, Internet Explorer 9, Opera 11.50, Apple Safari 5.1, Maxthon 3.0, SeaMonkey 2.1 – 2.6
2012	Google Chrome 17 – 23, Mozilla Firefox 10 – 17, Internet Explorer 10, Opera 12, Apple Safari 6, Maxthon 4.0, SeaMonkey 2.7-2.14
2013	Google Chrome 24 – 31, Mozilla Firefox 18 – 26, Internet Explorer 11, Opera 15 – 18, Apple Safari 7, SeaMonkey 2.15-2.23
2014	Google Chrome 32 – 39, Mozilla Firefox 27 – 34, Opera 19 – 26, Apple Safari 8
2015	Google Chrome 40 – 47, Microsoft Edge20 – 25, Mozilla Firefox 35 – 43, Opera 27 – 34, Vivaldi
2016	Google Chrome 48 – 55, Mozilla Firefox 44 – 50, Microsoft Edge 38, Opera 35 – 42, Apple Safari 10, SeaMonkey 2.24 – 2.30, Pale Moon 26.0.0[17], Pale Moon 27.0.0[18]

2017	Google Chrome 56 – 63, Microsoft Edge 40 – 41, Mozilla Firefox 51 – 57, Opera 43 – 49
2018	Google Chrome 64 – 71, Microsoft Edge 42 – 44, Mozilla Firefox 58 – 64, Opera 50 – 57
2019	Google Chrome 72 – 79, Microsoft Edge 48, Mozilla Firefox 65 – 71, Opera 58 – 65
2020	Google Chrome 80 – 87, Microsoft Edge 79 – 87, Mozilla Firefox 72 – 84, Opera 66 – 73
2021	Google Chrome 88 – 96, Microsoft Edge 88 – 96, Mozilla Firefox 85 – 95, Opera 74 – 82
2022	Google Chrome 97 – 108, Microsoft Edge 97 – 108, Mozilla Firefox 96 – 108, Opera 83 – 94
2023	Google Chrome 109, Microsoft Edge 109, Mozilla Firefox 109, Opera 94

(e) URLs

In order to allow clients to request particular resources from the server, a naming mechanism is required so that the client knows how to ask the server for the file. For the web, that naming mechanism is the Uniform Resource Locator (URL). A URL is the address of a document found on the WWW. Browser interprets the information in the URL in order to connect to the proper Internet server and to retrieve your desired document. Each time a click on a hyperlink in a WWW document instructs browser to find the URL that's embedded within the hyperlink.

The elements in a URL: **Protocol://server's address (domain)/filename**

Example of URLs with various different protocols:

1. Hypertext protocol: *http://www.mit.edu*
2. File Transfer Protocol: *ftp://ftp.dartmouth.edu*
3. Telnet Protocol: *telnet://pac.carl.org*
4. News Protocol: *news:alt.rock-n-roll.stones*

(f) How URL works

As illustrated in the figure below, a URL consists of two required components: the protocol used to connect, and the domain (or IP address) to connect to. Optional components of the URL are the path (which identifies a file or directory to access on that server), the port to connect to, a query string, and a fragment identifier.

i. Protocol

The first part of the URL is the protocol that we are using. Many of the application layer protocols of the TCP/IP stack can appear in a URL, and define what application protocols to use. Requesting *ftp://example.com/abc.txt* sends out an FTP request on port 21, while *http://example.com/abc.txt* would transmit on port 80.

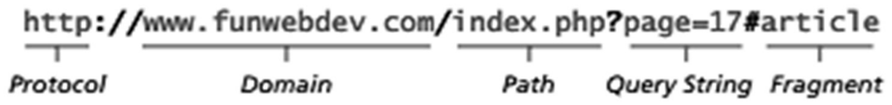


Figure 1 URL Components

ii. Domain

The domain identifies the server from which we are requesting resources. Since the DNS system is case insensitive, this part of the URL is case insensitive. Alternatively, an IP address can be used for the domain.

Domains divide World Wide Web sites into categories based on the nature of their owner, and they form part of a site's address, or uniform resource locator (URL). Common top-level domains are:

- ✓ .com – commercial enterprises
- ✓ .mil – military site
- ✓ .org – organization site (non-profits, etc.)
- ✓ .int – organizations established by international treaty
- ✓ .net – network
- ✓ .biz – commercial and personal
- ✓ .edu – educational site (universities, schools, etc.)
- ✓ .info – commercial and personal
- ✓ .gov – government organizations
- ✓ .name – personal sites

Additional three-letter, four-letter, and longer top-level domains are frequently added. Each country linked to the Web has a two-letter top-level domain, for example **.ng** is Nigeria, **.us** is United States, **.fr** is France, **.ie** is Ireland.

iii. Port

The optional port attribute allows us to specify connections to ports other than the defaults defined by the IANA authority. A port is a type of software connection point used by the underlying TCP/IP protocol and the connecting computer. If the IP address is analogous to a building address, the port number is analogous to the door number for the building.

Although the port attribute is not commonly used in production sites, it can be used to route requests to a test server, to perform a stress test, or even to circumvent Internet filters. If no port is specified, the protocol component of a URL determines which port to use. The syntax for the port is to add a colon after the domain, then specify an integer port number. Thus for instance, to connect to our server on port 888 we would specify the URL as *http://funwebdev.com:888/*.

iv. **Path**

The path is a familiar concept to anyone who has ever used a computer file system. The root of a web server corresponds to a folder somewhere on that server. On many Linux servers that path is /var/www/html/ or something similar (for Windows IIS machines it is often /inetpub/wwwroot/). The path is case sensitive, though on Windows servers it can be case insensitive. The path is optional. However, when requesting a folder or the top-level page of a domain, the web server will decide which file to send you. On Apache servers it is generally **index.html** or **index.php**. Windows servers sometimes use **default.html** or **default.aspx**. The default names can always be configured and changed.

v. **Query strings**

Query strings are the way of passing information such as user form input from the client to the server. In URLs, they are encoded as key-value pairs delimited by "&" symbols and preceded by the "?" symbol. The components for a query string encoding a username and password are illustrated in the figure below.

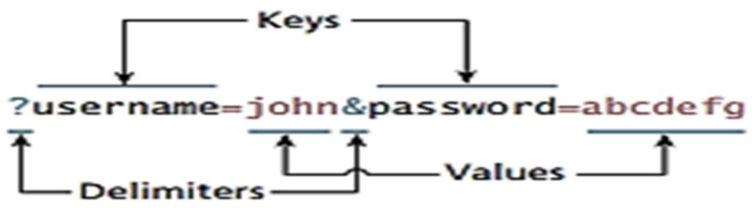


Figure 2 Query string components

vi. **Fragments**

The last part of a URL is the optional fragment. This is used as a way of requesting a portion of a page. Browsers will see the fragment in the URL, seek out the fragment tag anchor in the HTML, and scroll the website down to it. Many early websites would have one page with links to content within that page using fragments and "back to top" links in each section.

(g) HTTP

HTTP means HyperText Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed. HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input.

A similar abbreviation, HTTPS means Hyper Text Transfer Protocol Secure. Basically, it is the secure version of HTTP. Communications between the browser and website are encrypted by Transport Layer Security (TLS), or its predecessor, Secure Sockets Layer (SSL).

(h) Web Application Development

- **What is a Web Application?**

  A web application, also known as a web app, is a software program that runs on a web server and is accessed over the internet through a web browser. Web applications are designed to be accessible to users from any location, as long as they have an internet connection and a web browser.

  Web applications are built using a combination of web development technologies such as HTML, CSS, and JavaScript. These technologies are used to create the user interface and layout of the application, as well as to add interactive elements and dynamic functionality. Additionally, web applications can be built using various frameworks and libraries such as AngularJS, ReactJS, Vue.js, Ruby on Rails, etc. They are typically designed to be responsive, which means that they can adjust their layout and functionality based on the device being used to access them. This allows web apps to be used on a variety of devices, including desktops, laptops, tablets, and smartphones.

  Web applications are typically backed by a server-side technology like PHP, Python, Ruby, or Java. These technologies are used to handle the logic and data storage of the application, such as processing form submissions, managing user accounts, and storing data in a database.

Web applications are popular because they can be accessed by many users at once, and they can be updated and maintained easily. With the help of web applications, businesses can provide their services online, which results in cost-effective solutions, and higher scalability.

Simply put, Web Applications are dynamic web sites combined with server-side programming which provide functionalities such as interacting with users, connecting to back-end databases, and generating results to browsers.

Examples of web applications include: Online Shopping Platforms, Social Media Sites, Email Services, Project Management Tools, Registration Portals, Online Banking, Social Networking, Online Reservations, eCommerce/Shopping Cart Applications, Interactive Games, Online Training, Online Polls, Blogs, Online Forums, Content Management Systems and many more.

- **Static Websites versus Dynamic Websites**

In the earliest days of the web, a webmaster (the term popular in the 1990s for the person who was responsible for creating and supporting a website) would publish web pages and periodically update them. Users could read the pages but could not provide feedback. The early days of the web included many encyclopedic, collection style sites with lots of content to read (and animated icons to watch). In those early days, the skills needed to create a website were pretty basic: one needed knowledge of the HTML and perhaps familiarity with editing and creating images. This type of website is commonly referred to as a **static website**, in that it consists only of HTML pages that look identical for all users at all times.

Within a few years of the invention of the web, sites began to get more complicated as more and more sites began to use programs running on web servers to generate content dynamically. These server-based programs would read content from databases, interface with existing enterprise computer systems, communicate with financial institutions, and then output HTML that would be sent back to the users' browsers. This type of website is called a **dynamic website** because the page content is being created at run time by a program created by a programmer; this page content can vary from user to user.

So while knowledge of HTML was still necessary for the creation of these dynamic websites, it became necessary to have programming knowledge as well. And by the late 1990s, other knowledge and skills were becoming necessary, such as CSS, Javascript, PHP, Python, usability, and security.

- **Web Applications in Comparison to Desktop Applications**

Web application is a software based on the Web. This can refer to almost anything Web related, including a Web browser or other client software that can access the Web. It can refer to software that runs on Web sites or software that is stored on Web sites and downloaded to the user. Examples are social network applications like Facebook and Twitter, Online registration portals, video repositories like YouTube, online games etc.

Desktop application however is a software that is found on a local PC which simulates an object normally found on an office desktop, such as a calculator, notepad and appointment calendar. Examples include: Microsoft Office suite, Adobe PDF reader, Corel Draw, Video players, music players, offline games etc.

The user experience for a web application is unlike the user experience for traditional desktop software. The location of data storage, limitations with the user interface, and limited access to operating system features are just some of the distinctions. However, as web applications have become more and more sophisticated, the differences in the user experience between desktop applications and web applications are becoming more and more blurred.

There are a variety of advantages and disadvantages to web-based applications in comparison to desktop applications. Some of the advantages of web applications include:

- ✓ Accessible from any Internet-enabled computer.
- ✓ Usable with different operating systems and browser applications.
- ✓ Easier to roll out program updates since only software on the server needs to be updated and not on every desktop in the organization.
- ✓ Centralized storage on the server means fewer security concerns about local storage (which is important for sensitive information such as health care data).

Unfortunately, in the world of IT, for every advantage, there is often a corresponding disadvantage; this is also true of web applications. Some of these disadvantages include:

- ✓ Requirement to have an active Internet connection (the Internet is not always available everywhere at all times).
- ✓ Security concerns about sensitive private data being transmitted over the Internet.
- ✓ Concerns over the storage, licensing, and use of uploaded data.
- ✓ Problems with certain websites on certain browsers not looking quite right.

- ✓ Restrictions on access to the operating system can prevent software and hardware from being installed or accessed.

(i) Web Application Development Explained

Web application development refers to the process of creating software applications that run on a web server, rather than on a local computer or mobile device. These applications are accessed via a web browser and can be used by anyone with an internet connection. They can range from simple, single-page apps to complex, multi-functional platforms that can handle millions of users. Web application development requires knowledge of web development technologies such as HTML, CSS, JavaScript, and server-side languages such as PHP, Ruby, and Python.

The process of web application development typically involves several stages, including:

i. **Planning and requirements gathering:** This is the initial stage where the project team determines the goals and objectives of the application, as well as the features and functionality that will be required. This stage also involves creating a project plan and identifying any potential constraints or challenges that may arise.

ii. **Design and user experience:** In this stage, the team creates the overall look and feel of the application, including the layout, color scheme, and branding. The team also considers the user experience, including how users will interact with the application and how the application will be organized to make it easy to use.

iii. **Development:** This is where the actual coding and programming of the application takes place. The team uses web development technologies such as HTML, CSS, JavaScript, and server-side languages such as PHP, Ruby, and Python to create the application. The team also integrates any necessary libraries, frameworks, or APIs to add additional functionality.

iv. **Testing and quality assurance:** Before the application is released, it must be thoroughly tested to ensure that it is free of bugs and that it functions as intended. This stage also includes user acceptance testing, where a group of users is brought in to test the application and provide feedback.

v. **Deployment and maintenance:** Once the application has been tested and is ready for release, it is deployed to a web server and made available to users. After the application is deployed, the team is responsible for maintaining it, including making updates and addressing any issues that arise.

Web Application development includes Frontend frameworks like Bootstrap, Angular, React, Vue.js, etc. Backend frameworks like Laravel, CakePHP, Zend, Express.js, Django, Ruby on Rails, etc.

Web Application development also includes security aspect like SSL certificate, JWT, OAuth, etc.

Overall, web application development is a complex and dynamic process that requires a variety of skills and a team of experienced developers to successfully create and maintain a web application.

(j) Web Application Development Technologies

There are two main categories of coding, scripting and programming for creating Web Applications, viz:

1. **Client-Side Scripting/Coding:** Client-Side Scripting is the type of code that is executed or interpreted by browsers. Client-Side Scripting is generally viewable by any visitor to a site (from the view menu click on "View Source" to view the source code). Below are some common Client-Side Scripting technologies:
    - HTML (HyperText Markup Language).
    - CSS (Cascading Style Sheets).
    - JavaScript
    - Ajax (Asynchronous JavaScript and XML).
2. **Server-Side Scripting/Coding:** Server-Side Scripting is the type of code that is executed or interpreted by the web server. Server-Side Scripting is not viewable or accessible by any visitor or general public. Below are the common Server-Side Scripting technologies:
    - PHP (very common Server-Side Scripting language - Linux/Unix based Open Source - free redistribution, usually combines with MySQL database).
    - ASP (Microsoft Web Server (IIS) Scripting language).
    - Perl (general purpose high-level programming language and Server Side Scripting Language - free redistribution - lost its popularity to PHP).
    - Python (general purpose high-level programming language and Server Side Scripting language - free redistribution).

(k) Program Libraries

Program libraries are a collection of commonly used functions, classes or subroutines which provide ease of development and maintanance by allowing developers to easily add or edit functionalities to a frameworked or modular type application.

(l) Web Application Frameworks

Web Application Frameworks are sets of program libraries, components and tools organized in an architecture system allowing developers to build and maintain complex web application projects using a fast and efficient approach. Web Application Frameworks are designed to streamline programming and promote code reuse by setting forth folder organization and structure, documentation, guidelines and libraries (reusable codes for common functions and classes). Below are some example of frameworks:

- **Bootstrap** (Free and Open-Source CSS Framework directed at responsive, mobile-first front-end web development).
- **JQuery** (JavaScript front-end Library designed to minimize the tedious Javascript coding and provide simplicity).
- **ReactJS** (Facebook's Open-Source JavaScript component-based Library that features JSX syntax).
- **AngularJS** (Google's Open-Source TypeScript Library that features JSX syntax
- **Zend** (PHP's Object Oriented Web Application Framework).
- **Django** (High-level Python web framework that enables rapid development of secure and maintainable websites).
- **ASP.NET** (Microsoft's Web Application Framework - successor of ASP).
- **ColdFusion** (Adobe's Web Application Framework).
- **Ruby on Rails** (Ruby programming's Web Application Framework - free redistribution).

(m) Web Application Frameworks - Benefits and Advantages

- Program actions and logic are separated from the HTML, CSS and design files. This helps designers (without any programming experience) to be able to edit the interface and make design changes without help from a programmer.
- Builds are based on the module, libraries and tools, allowing programmers to easily share libraries and implement complex functionalities and features in a fast and efficient manner.
- The structure helps produce best practice coding with consistent logic and coding standards, and provides other developers the ability to become familiar with the code in a short time.

(n) Coding Guidelines, Standards & Convention

Coding guidelines are sets of rules and standards used in programming a web application project. These rules and standards apply to coding logic, folder structure and names, file names, file organization, formatting and indentation, statements, classes and functions, and naming conventions. These rules also enforce writing clear comments and provide documentation. Important benefits of using Coding Guidelines:

- Creates the best environment for multiple programmers to work on the same project.
- Provides ease of maintainability and version management.
- Delivers better readability and understanding of the source code.
- Insures that other developers can understand and become familiar with the code in a short time.

(o) Web Applications Lifecycle Model

Web Application Lifecycle is the process of developing a web application and involvement of the multiple teams that are engaged in the development process. Each organization may set forth its own unique style of operating. Some companies follow a certain standard model such as SDLC (System Development Life Cycle) or Agile Software Development Model.

- SDLC is the traditional process of developing software or web applications by including research to identify and define the application requirements, information analysis, architectural design and specifications blueprint, team involvement, programming, testing and bug fixing, system testing, implementation and maintenance.
- Agile Software/Web Application Development is the iterative development process and development process practices that focus on collaboration of people involved and provide a better procedure to allow revisions and evolution of web application requirements. Agile methodology includes research, analysis, project management, design, programming, implementation, frequent testing, adaptation and maintenance.

(p) Web Application Development Process

Web Application Development Process organizes a practical procedure and approach in application development. The following list of procedures and suggested documents provide a good outline for a Web Application Lifecycle and Process:

- Roadmap Document: Defining Web Application, Purpose, Goals and Direction.
- Researching and Defining Audience Scope and Security Documents.
- Creating Functional Specifications or Feature Summary Document.
- Team Collaboration and Project Management Document.

- Technology Selection, Technical Specifications, Illustrative Diagram of Web Application Architecture and Structure, Development Methodology, Versions Control, Backups, Upgrades, Expansion and Growth Planning Document, Server Hardware / Software Selection.
- Third Party Vendors Analysis and Selection (Merchant Account and Payment Gateway, SSL Certificate, Managed Server/Colocated Server Provider, Fulfillment Centers, Website Visitor Analytics Software, Third Party Checkout Systems, etc.)
- Application Visual Guide, Design Layout, Interface Design, Wire Framing.
- Database Structure Design and Web Application Development.
- Testing: Quality Assurance, Multiple Browser Compatibility, Security, Performance - Load and Stress Testing, Usability.
- Maintenance

(q) Web Application Testing

Testing is an important part of the Web Application Development process. On occasion, testing would consume more manpower and time than development itself. Below are some of the most common testing needed for any web application development process:

- Quality Assurance and Bug Testing.
- Multiple Browser Compatibility.
- Application Security.
- Performance - Load and Stress Testing.
- Usability.

(r) Trends and Popularity

The demands for companies to build Web Applications are growing substantially. If planned and built correctly, web applications can:

- Reach and service millions of consumers and businesses.
- Generate substantial, multi-layer/multi-category income from consumers, businesses and advertisers.
- Easily build business goodwill and assets based on audience reach, popularity, technology and potential growth

Below are good reasons for companies to build web applications:

- Companies want to streamline their internal departments and functions, operations, sales and project management, etc.
- Companies want to take advantage of a web based application's flexibility and versatility, by moving away from the traditional desktop application platform to the web application platform.
- Companies want to gain more clients or better service their current clients by offering convenient services and solutions online.
- Companies want to build new web applications to offer innovative services or solutions to online users and businesses

(s) Business Impact

Today's web applications have substantial business impact on the way companies and consumers do business such as:

- There are opportunities to gain the upper hand and bypass the traditional brick and mortar companies when this type of opportunity was rarely possible or existed before the explosion of the web.
- The new web created a global business environment which challenges the way in which traditional companies do business.
- Companies need to reinvent and evolve in order to compete in today's trends, online business and global marketplace.
- Businesses and consumers have more options and resources to research and easily compare and shop around for the best deals.
- Information and resources are immense and available to everyone who seeks it.
- Businesses or companies who used to profit from consulting or advice, that can now be easily acquired online are struggling, and will need to take a new business direction if they want to stay solvent.

UNIT – II

Frontend (Client-side) Development: Introduction to Frontend Web Development Using Frontend Web Technologies and Frameworks

(a) HTML

HTML, an acronym for Hypertext Markup Language, is a computer markup scripting language (not a programming language) for creating websites and web applications. Consisting mainly series of codes usually written in a text file and saved as html or htm file, code written in the HTML language translates into a beautiful, well-formatted text or a combination of text and media when viewed through a browser. Unlike a scripting or programming language that uses scripts to perform functions, a markup language uses tags to identify content.

HTML was first developed by Tim Berners-Lee in 1990, and it has gone through so many evolutions since then that the most recent version can achieve far more than was imagined possible when the language was first invented. HTML is very important in Web development because it used to build the structure of the Web page and to output information via the browser.

(i) HTML Versions

First, a quick rundown of all the HTML versions since HTML was invented.

- **HTML 1.0**: This was the barebones version of HTML and the very first release of the language.
- **HTML 2.0**: This version was introduced in 1995. It gradually evolved, allowing extra capabilities including form-based file upload, tables, client-side image maps and internationalization.
- **HTML 3.2**: In an attempt to ensure development of standards for the World Wide Web, the World Wide Web Consortium (W3C) was founded by Tim Berners-Lee in 1994. By 1997, they published HTML 3.2.
- **HTML 4.0**: Later in 1997, the W3C released HTML 4.0 — a version that adopted many browser-specific element types and attributes. HTML 4.0 was later reissued with minor edits in 1998.
- **HTML 4.01**: In December 1999, HTML 4.01 was released.
- **XHTML**: The specifications were introduced in 2000 and it was recommended to be used as the joint-standard with HTML 4.01. It incorporated XML to ensure code is properly written and to ensure interoperability between programming languages.

- **HTML5**: The W3C published HTML5 as a recommendation in October 2014 and later released HTML 5.1 in November 2016.

(ii) Writing your first HTML Web page

If you are thinking of starting to create web pages in HTML, then all you need is a text editor and you are good to go. Notepad is a simple text editor which comes preinstalled in Windows PC and will do the job for you. Web pages can also be created and modified by using professional HTML editors or integrated development environments (IDEs). However, for learning HTML it is recommended to use a simple text editor like Notepad (PC) or TextEdit (Mac). It is believed that using a simple text editor is a good way to learn HTML. However, there are of course several benefits to using a professional HTML editor or an IDE.

A good HTML editor will keep your code clean and organized. It will also detect when you open a new tag and automatically close it to avoid you having a buggy code and as a result reducing how much typing you have to do. Most HTML editors/IDEs today allow you to preview your web page to see how it will look like in a web browser using their WYSIWYG feature.

There are many free and paid HTML editors, below are some of the top options you can choose from:

- Notepad
- Dreamweaver
- CoffeeCup
- KompoZer
- Komodo Edit
- Notepad++
- Bluefish
- VS Code
- Sublime text
- Brackets

The following four steps below will show you how to create your first web page with Notepad.

Step 1 - Open Notepad (PC):

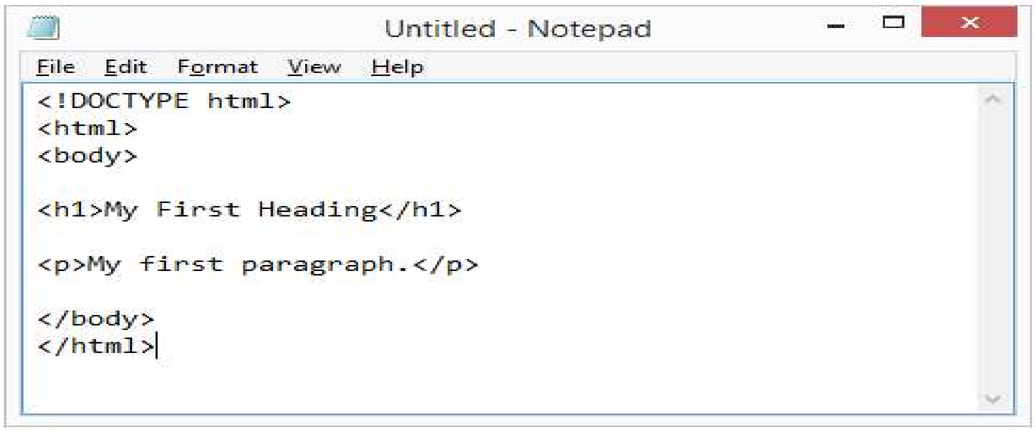
- Windows 8:

Open the Start Screen (the window symbol at the bottom left on your screen). Type Notepad.

- Windows 7 or 10:

  Open Start > Programs > Accessories > Notepad

Step 2 - Write some HTML script into Notepad.

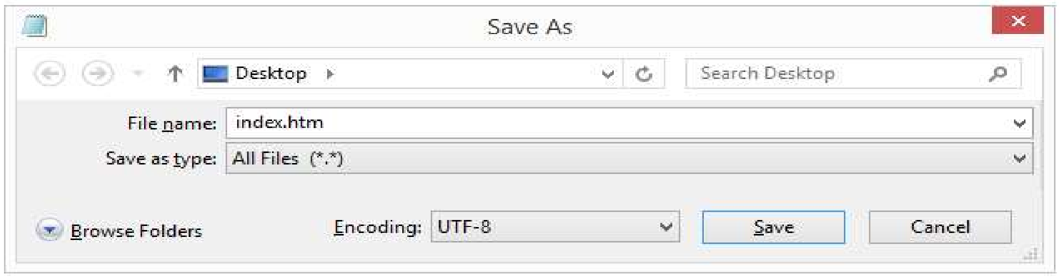


```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Figure 3 Writing HTML in Notepad

Step 3 - Save the HTML Page:

- Save the file on your computer. Select File > Save as in the Notepad menu.
- Name the file "index.html" and set the encoding to UTF-8 (which is the preferred encoding for HTML files).
- You can use either **.htm** or **.html** as file extension. There is no difference, it is all the same.

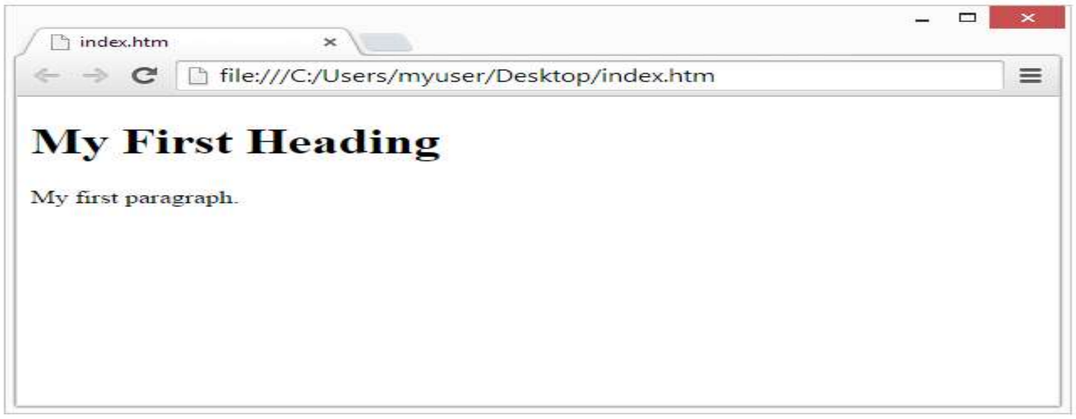


File name: index.htm
Save as type: All Files (*.*)
Encoding: UTF-8

Figure 4 Saving HTML file

Step 4 - View the HTML Page in Your Browser:

- Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").
- The result will look much like this:



My First Heading

My first paragraph.

Figure 5 View HTML page in Browser

(iii) HTML Syntax

The W3C Recommendation for HTML specifies the syntax for marking up documents. It was defined and centred on using three components namely: tags, elements and attributes. Learning the fundamental concepts and terms that have survived multiple standards is essential in a discipline like Web Application Development where specifications, standards, and browsers are constantly evolving.

(iv) HTML Tags, Elements and Attributes

When building websites it's important to understand what tags, elements attributes and their values are. Once you understand what they are and how they all relate to one another, it's relatively simple to create good and valid HTML.

a) HTML Tags:

   Tags are used to define the beginning and end of an HTML element. They consist of an opening bracket (<), followed by the name of the tag and then a closing bracket (>). If an attribute is being used in the tag, it will be included after the tag.

   Here’s an example of the HTML title tag:

```
<title> </title>
```

b) HTML Elements:

   The element consists of both the opening and closing tags as well as what’s inside those tags. It normally consists of some structure that’s used to define the respective tags.

   Here’s an example of the HTML title element:

```
<title>My Webpage</title>
```

   In this example, the HTML element is a title of “My Webpage,” complete with the opening and closing title tags.

c) HTML Attributes:

Attributes are used to define a property for one or more HTML elements. They are found within the element's opening tag, often containing spaces that are separated by value pairs. An HTML attribute can also be explained as a name = "value" pair that provides more information about the HTML element. In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional, though many web authors still maintain the practice of enclosing attribute values in quotes. Some HTML attributes expect a number for the value. These will just be the numeric value; they will never include the unit.

Here's are some examples:

- HTML background color attribute:
  `<body bgcolor = "blue">...</body>`
- HTML align text attribute:
  `<p align = "center">This is a paragraph</p>`
- HTML alt text attribute:
  `<img src="dogs-playing.gif" alt="dogs playing outside">`

(v) Basic Structure of HTML Document

Within the Web page example below, tags were used to markup the structure of the content with opening tags and closing tags. The page was started off with an `<html>` opening tag and finished with an `</html>` closing tag. Within this tag other tags were also nested to build the page.

It's best to remember that all tags must be closed (there can be some exceptions depending on which doctype you're using but it's best when initially starting out to close all tags). Where a tag

```
<!DOCTYPE html> <!-- tells browser above the html version -->
<html>
      <!-- beginning of the html document -->
      <head>
      <!-- header related tags e.g. title, links etc. -->
      </head>
      <body>
      <!-- actual html document here -->
      <h1>Main heading goes here</h1>
      <p>A paragraph goes here</p>
      <!-- add JavaScript files here -->
      </body>
</html>
```

Code 1 Basic Structure an HTML5 document doesn't contain any content, the tag is said to be self-closing. For example the line-break tag `<br>` and the image tag `<img>`.

Explanation of HTML code above:

- Entries inside the /< ... /> are known as tags. Most of the tags has an opening and closing e.g. <head>(opening head) and </head> (closing head). Some of the tags do not have closing tags e.g. <!DOCTYPE ...> and <br />. We need to write the HTML codes inside the tags.
- The comments are written between '<!–' and '–>'.
- Here Line 1 gives the details of the 'HTML version' to the web-browser. The 'html' tells it is version 5.
- The 'head' tag (Lines 4-6) contains the header related tags e.g. 'title for the page' and 'links for the css files'etc.
- The 'body' tag (7-12) contains the actual HTML code which is displayed on the web-browser. Also, we add all the JavaScript related codes just before the closing body tag (</body>).

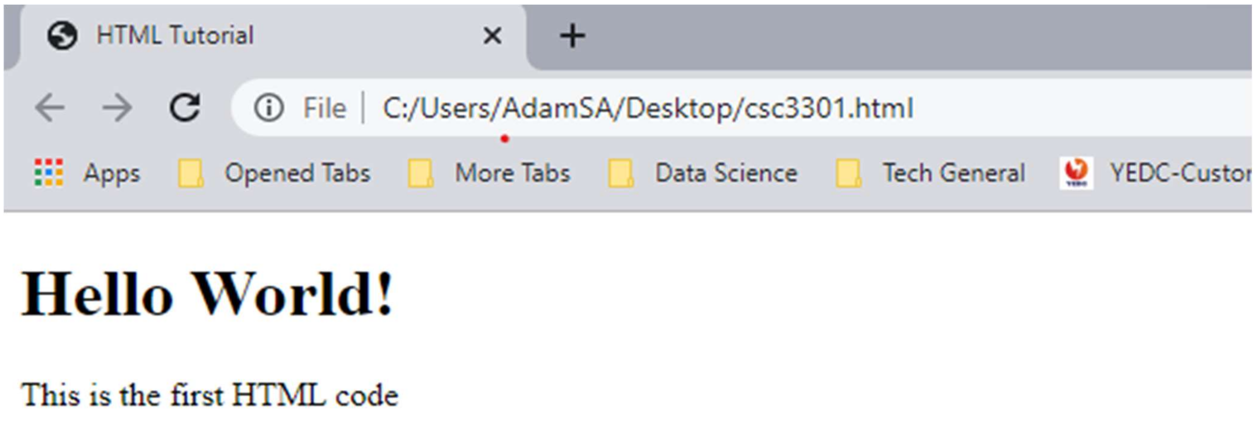
(vi) HTML Example

In below code, the message "Hello World" is displayed on the HTML page. The Figure 6 is the resultant HTML page.

- The title (Line 4) appears on the top of the browser.
- The tag <h1> is called 'header' tag, which has the larger size than the normal text (see the size of 'Hello World!').
- The tag <p> is called the 'paragraph' tag, which can be used to write the paragraphs.

```
1     <!DOCTYPE html>
2     <html>
3           <head>
4                 <title>HTML Tutorial</title>
5           </head>
6           <body>
7                 <h1> Hello World! </h1>
8                 <p> This is the first HTML code </p>
9           </body>
10    </html>
```

Code 2 HTML Example 1



Hello World!

This is the first HTML code

Figure 6 HTML Example 1 Output

(vii) Basic Tags

The Table 1 shows the list of tags which are required for writing the basic 'HTML' codes i.e. without any style e.g. bold, italics and numbering etc.

Table 1 List of basic tags

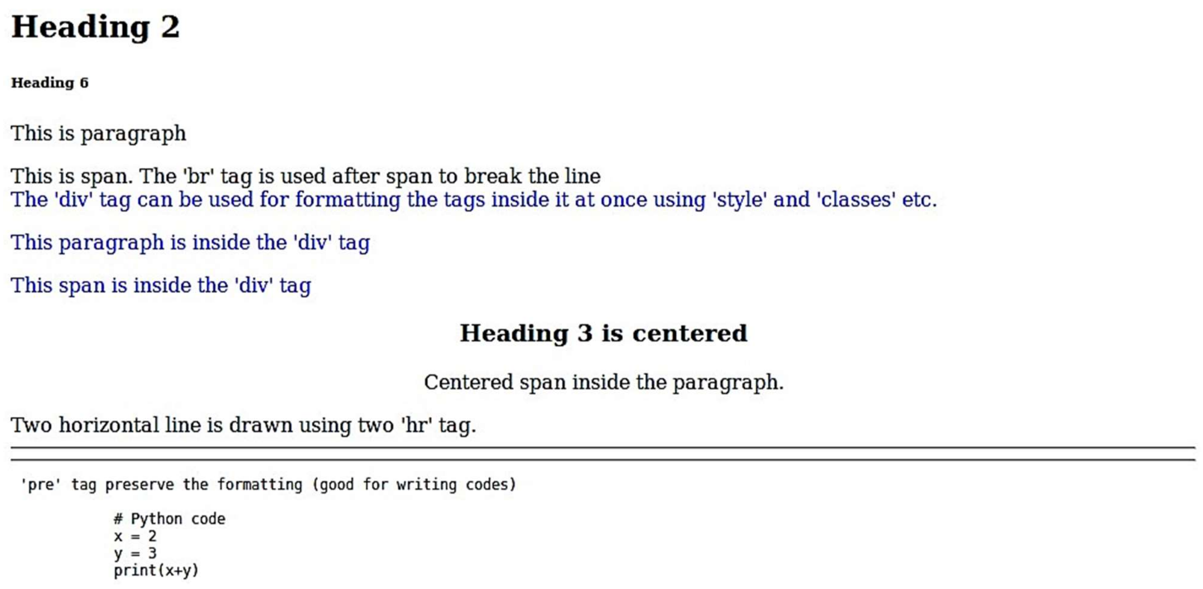
Tag	**Description**	**Example**
h1, ..., h6	Header tag h1 to h6	<h2> Hi </h2>
P	paragraphs (Line changes at the end)	<p> Hi </p>
Span	No line change after span	<span>Hi</span> Bye.
Div	make division between contents	<div> ... </div>
A	Hyperlink	<a>Click Here</a>
Center	Move content to center	<center> Hi </center>
Br	Line break (no closing tag)	<br /> or <br>
Hr	horizontal line (no closing tag)	<hr /> or <hr>
Pre	preserve formatting	<pre> .... </pre>
Table	insert table	

Let's see the example of each of these tags.

***Note:** All the new codes are added below the previous codes in **Error! Reference source not found.Error! Reference source not found.** in the 'body' tag. Therefore, only newly added codes are*

```
<h2> Heading 2 </h2>
      <h6> Heading 6 </h6>

      <p> This is paragraph </p>
      <span> This is span.</span>
      <span> The 'br' tag is used after span to break the line </span>
      <br/>

      <div style="color:blue;">
      The 'div' tag can be used for formatting the tags inside it at once using
      'style' and 'classes' etc.
            <p> This paragraph is inside the 'div' tag </p>
            <span> This span is inside the 'div' tag </span>
            <br/>
      </div>

      <center>
            <h3> Heading 3 is centered</h3>
            <p><span> Centered span inside the paragraph.</span><p>
      </center>

      Two horizontal line is drawn using two 'hr' tag.
      <hr />
      <hr>

      <pre> 'pre' tag preserve the formatting (good for writing codes)

            # Python code
            x = 2
            y = 3
            print(x+y)

      </pre>
```

Code 3 HTML Example 2



Heading 2

Heading 6

This is paragraph

This is span. The 'br' tag is used after span to break the line
The 'div' tag can be used for formatting the tags inside it at once using 'style' and 'classes' etc.

This paragraph is inside the 'div' tag

This span is inside the 'div' tag

Heading 3 is centered

Centered span inside the paragraph.

Two horizontal line is drawn using two 'hr' tag.

```
'pre' tag preserve the formatting (good for writing codes)

          # Python code
          x = 2
          y = 3
          print(x+y)
```

Figure 7 HTML Example 2 Output : Attribute 'style' is used in 'div' tag

(viii) Attributes

In Figure 7, we saw an example of attribute (i.e. style) which changed the color of all the elements to 'blue' inside the 'div' tag.

1) Attribute 'name' and 'value'

- Attribute is defined inside the opening part of a 'tag'. For example, in the below code, the attribute 'style'is defined inside the 'div' tag.

```
<div style="color:blue;">

</div>
```

- An attribute has two parts i.e. 'name' and 'value'. For example, in the above code, name and value of the attribute are 'style' and 'blue' respectively.

2) Core attributes

Below are the three core attributes which are used frequently in web design.

- **id:** The 'id' is the unique name which can be given to any tag. This is very useful in distinguishing the element with other elements.

```
<p id='para1'> This is paragraph with id 'para1' </p>
<p id='para2'> This is paragraph with id 'para2' </p>
```

- **class:** The attribute 'class' can be used with multiple tags. This is very useful in making groups in HTML design.

```
<p class="c_blue"> This is paragraph with class 'blue'</p>
<span class="c_blue"> This is span with class 'blue'</span>
```

- **style:** We already see the example of style attribute, which can be used to change the formatting of the text in HTML design. We can specify various styles which are discussed in CSS.

```
<p style="font-weight:bold; color:red;">Style attribute is used to
bold and color</p>
```

Note: *Above three attributes are used with 'CSS (cascading style sheet)' and JavaScript/jQuery, which are the very handy tools to enhance the look and functionalities of the web-page respectively.*

- Also we can define multiple attributes for one tag as shown below,

```
<p class="my_class" id="para_with_class" style="color:green">
Multiple attributes </p>
```

- The other useful attributes are listed in Table 2 below:

Table 2 List of Attributes

Name	**Values**	**Description**
Id	user defined names	<p id='p_1'> Hi </p>
Class	user defined names	<p class='p_class'> Hi </p>
Style	CSS styles	<p style="color:red; font-weight:bold;"> Hi </p>
Align	left, right, center	horizontal alignment
Width	numeric value or % value	width of images and tables etc.
Height	numeric value	height of images and tables etc.

(ix) Tables

Here, we will learn to draw tables along with some attributes which are discussed in Table 2. Table 3 shows the list of tags available to create the table, which are used in Code 4.

Table 3 Tags and Attributes for making Table

Tag	**Description**
table	beginning and end of table
tr	row of table
th	header cell

td	data cell
Attributes	**Description**
Rowspan	number of rows to merge
Colspan	number of columns to merge
Border	width of border
cellpadding	width of whitespace between two border
cellspacing	width of whitespace within a border
Bgcolor	background color
bordercolor	color of border
Width	width of table (numeric or %)
Height	height of table (numeric)
Caption	caption for table

Some of the attributes of Table 3 are used in below example,

```
<!-- border-color, width and height -->
<table border="1" bordercolor="black" width="450" height="100">
<caption>Table 1 : Various tags of table</caption>
   <tr bgcolor="red" > <!-- row -->
      <th>Column 1</th> <!-- header -->
      <th>Column 2</th>
      <th>Column 3</th>
   </tr>
   <tr bgcolor="cyan"> <!-- background color -->
      <td>Data 1</td> <!-- data -->
      <td>Data 2</td>
      <td>Data 3</td>
   </tr>
   <tr bgcolor="yellow"> <!-- row -->
      <td colspan="2">New Data 1</td> <!-- column span -->
      <td>New Data 2</td> <!-- data --> </tr>
</table>
<!-- width in % -->
<table border="1" bordercolor="black" width="80%" height="100">
<caption> Table 2 : Width is 80%</caption>
   <tr bgcolor="red" >
      <th>Column 1</th>
      <th>Column 2</th>
     <th>Column 3</th>
   </tr>
   <tr bgcolor="cyan"> <!-- row -->
      <td>Data 1</td> <!-- data -->
      <td>Data 2</td>
      <td>Data 3</td>
   </tr>
</table>
```

Code 4 HTML Example 3

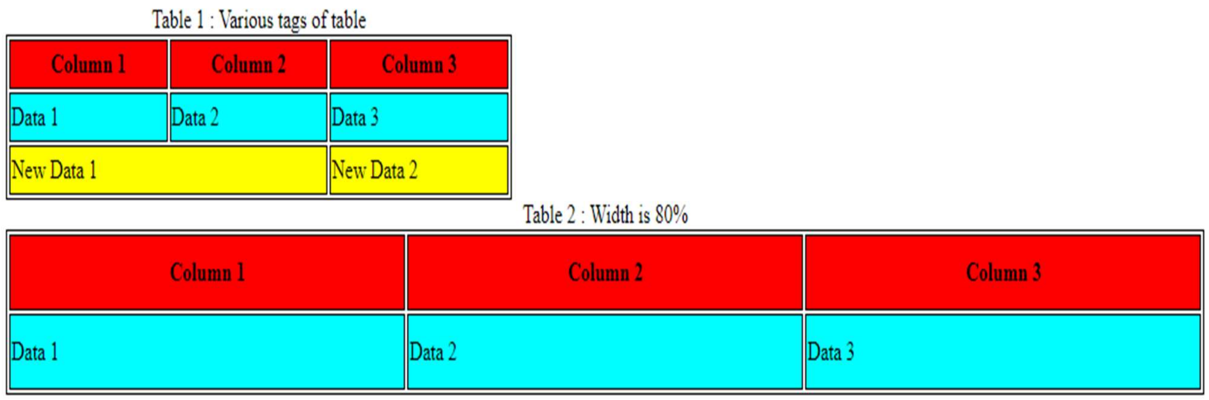


Table 1 : Various tags of table

Column 1	Column 2	Column 3
Data 1	Data 2	Data 3
New Data 1		New Data 2

Table 2 : Width is 80%

Column 1	Column 2	Column 3
Data 1	Data 2	Data 3

Figure 8 HTML Example 3 Output

(x) Text formatting

Some of the text formatting options are shown in Table 4 e.g. bold, italic, subscript and strike etc.

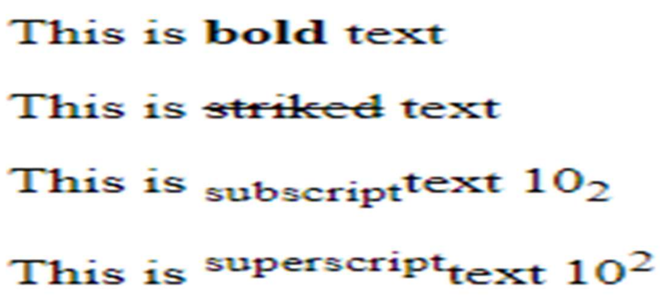
Table 4 Text Formatting Tags

Tag	Description
b	Bold
i	Italic
u, ins	Underline
strike, del	Strike
sup	Superscript
sub	Subscript
big	big size text
small	small size text

Below are the some of the examples of text formatting, whose results are shown in Figure 9.

```
<!-- Text formatting -->
<p>This is <b>bold</b> text</p>
<p>This is <strike>striked</strike> text</p>
<p>This is <sub>subscript</sub>text 10<sub>2</sub></p>
<p>This is <sup>superscript</sup>text 10<sup>2</sup></p>
```

Code 5 HTML Example 4


This is **bold** text

This is ~~striked~~ text

This is $_{\text{subscript}}$text 10_2

This is $^{\text{superscript}}$text 10^2

Figure 9 HTML Example 4 output: Text Formatting

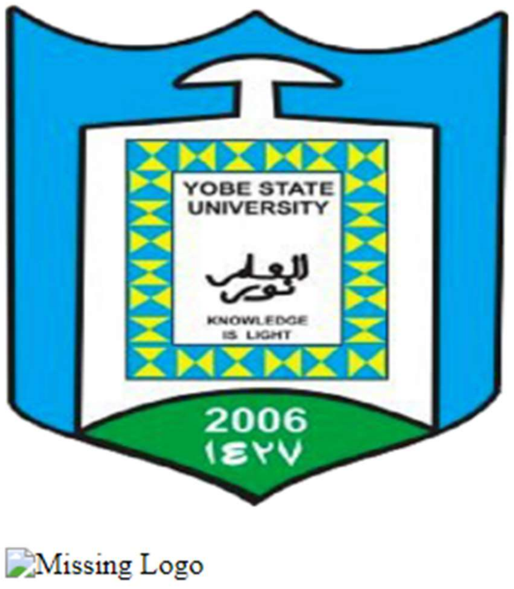
(xi) Images

The image tag <img> is a self closing and has two important attribues i.e. 'src' and 'alt' as described below:

- src: tells the location of 'image' file e.g. the image 'logo.jpg' will be searched inside the folder 'img'.
- alt: is the 'alternate text' which is displayed if image is not found. For example, the name of the image is incorrectly written i.e. 'logoa' (instead of 'logo'), therefore the value of 'alt' i.e. 'Missing Logo' will be displayed as shown in Fig. 1.5.

```html
<!-- Images -->
<img src="img/logo.jpg" alt="YSU Logo.jpg" width="20%">
<br/> <br/>
<img src="img/logoa.jpg" alt="Missing Logo.jpg" width="20%">
```

Code 6 HTML Example 5



Missing Logo

The required src attribute specifies the URL of the image. There are two ways to specify the URL in the src attribute:

1. Absolute URL - Links to an external image that is hosted on another website. Example: src="https://www.ysu.edu.ng/images/logo.jpg".
2. Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current

page. Example: src="logo.jpg". If the URL begins with a slash, it will be relative to the domain. Example: src="/images/logo.jpg".

It is almost always best to use relative URLs. They will not break if you change domain.

(xii) Links

HTML links are also called hyperlinks. Clicking on a link jumps the reader to another document. When the mouse is moved over a link, the mouse arrow will turn into a little hand.

Note: *A link does not have to be text. A link can be an image or any other HTML element!*

The HTML `<a>` tag defines a hyperlink and it has the following syntax:

```
<a href="url">link text</a>
```

The most important attribute of the `<a>` element is the ***`href`*** attribute, which indicates the link's destination. The link *text* is the part that will be visible to the reader. Clicking on the link text, will send the reader to the specified URL address.

By default, links will appear as follows in all browsers:

- An **unvisited** link is underlined and **blue**
- A **visited** link is underlined and **purple**
- An **active** link is underlined and **red**

Links can be styled with CSS, to give it another look. By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

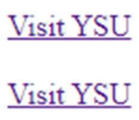
The **`target`** attribute specifies where to open the linked document. The `target` attribute can have one of the following values:

- **`_self`** - Default. Opens the document in the same window/tab as it was clicked.
- **`_blank`** - Opens the document in a new window or tab.
- **`_parent`** - Opens the document in the parent frame.
- **`_top`** - Opens the document in the full body of the window

Both examples in Code 7 below are using an **absolute URL** (a full web address) in the *href* attribute.

```
<!-- links -->
<a href="https://ysu.edu.ng/"> Visit YSU</a><br>
<a href="https://ysu.edu.ng/" target="_blank">Visit YSU</a>
```

Code 7 HTML Example 6


Visit YSU

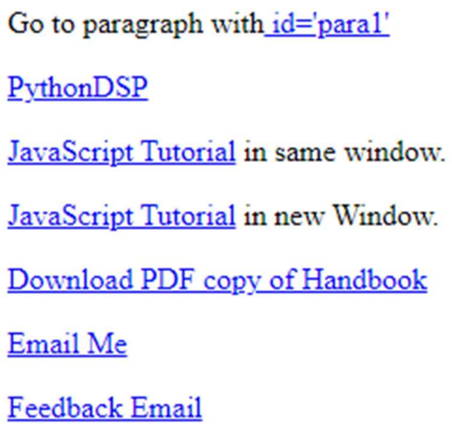
Visit YSU

Figure 10 HTML Example 6 Output

A local link (a link to a page within the same website) shown in is specified with a **relative URL** (without the "https://www" part). A link can also be made to a part of a page using the ‘id’ attribute aswell.

```
<!--more links -->
<p>Go to paragraph with<a href="#para1"> id='para1'</a></p>
<a href="http://pythondsp.readthedocs.io"> PythonDSP </a>
<br>
<p><a href="js.html" target="_self"> JavaScript Tutorial</a> in same window.</p>
<p><a href="js.html" target="_blank"> JavaScript Tutorial</a> in new Window.</p>
<p><a href="http://ysu.edu.ng/student.pdf">Download PDF copy of Handbook</a></p>
<p><a href="mailto:adamsa@gmail.com">Email Me</a></p>
<p><a href="mailto:info@ysu.edu.ng.com?subject=Feedback&body=Your feedback
here">Feedback Email</a></p>
```

Code 8 HTML Example 7


Go to paragraph with id='para1'

PythonDSP

JavaScript Tutorial in same window.

JavaScript Tutorial in new Window.

Download PDF copy of Handbook

Email Me

Feedback Email

Figure 11 HTML Example 7 Output

***Note:** We can change the color of the links using 'alink (active link)', 'link' and 'vlink (visited link', by defining these attributes in the 'body tag' as shown below,*

```
<body alink="green" link="blue" vlink="red">
```

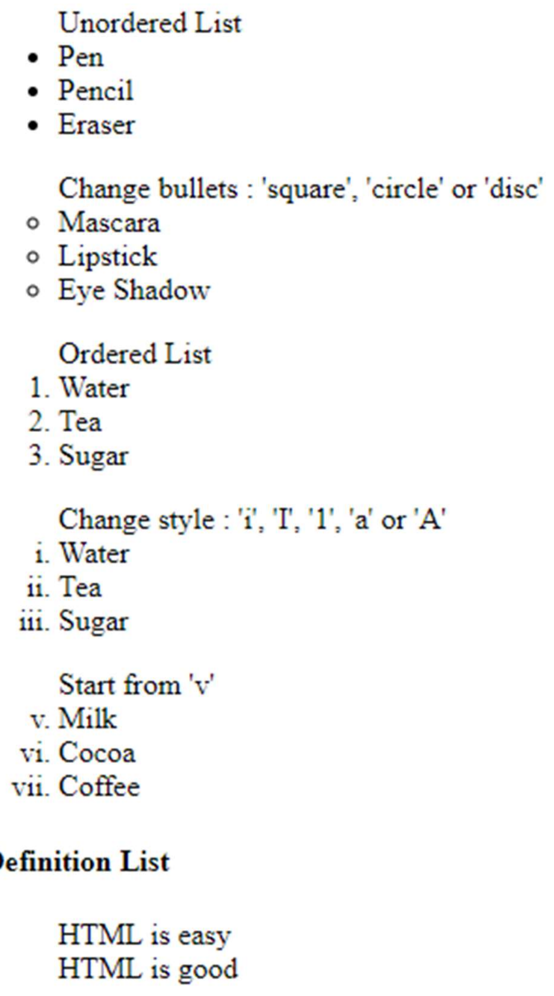
(xiii) Lists

There are three type of lists in HTML,

- Unordered list : bullet are used in it.
- Ordered list : numbers are used in it.
- Definition list : This can be used for writing definitions.

```
<!-- Lists -->
      <!-- unordered list -->
      <ul> Unordered List
            <li>Pen</li>
            <li>Pencil</li>
            <li>Eraser</li>
      </ul>
      <ul type="circle"> Change bullets : 'square', 'circle' or 'disc'
            <li>Mascara</li>
            <li>Lipstick</li>
            <li>Eye Shadow</li>
      </ul>

      <!-- ordered list -->
      <ol> Ordered List
            <li>Water</li>
            <li>Tea</li>
            <li>Sugar</li>
      </ol>
      <ol type='i'> Change style : 'i', 'I', '1', 'a' or 'A'
            <li>Water</li>
            <li>Tea</li>
            <li>Sugar</li>
      </ol>
      <ol type='i' start="5"> Start from 'v'
            <li>Milk</li>
            <li>Cocoa</li>
            <li>Coffee</li>
      </ol>

      <!-- Definition list -->
      <dl>
            <dt>
                  <h4>Definition List</h4>
            </dt>
            <dd> HTML is easy </dd>
            <dd> HTML is good </dd>
      <dl>
```

Code 9 HTML Example 8



```
     Unordered List
  •  Pen
  •  Pencil
  •  Eraser

     Change bullets : 'square', 'circle' or 'disc'
  o  Mascara
  o  Lipstick
  o  Eye Shadow

     Ordered List
  1. Water
  2. Tea
  3. Sugar

     Change style : 'i', 'I', '1', 'a' or 'A'
   i. Water
  ii. Tea
 iii. Sugar

     Start from 'v'
    v. Milk
   vi. Cocoa
  vii. Coffee

Definition List

     HTML is easy
     HTML is good
```

Figure 12 HTML Example 7 Output

(xiv) Forms

HTML form is used to collect user input. The user input is most often sent to a server for processing.

The HTML <form> element is used to create an HTML form for user input:

<form>
.
form elements
.
</form>

Forms can have different types of controls to collect the data from users, which are listed below and shown in Table 5:

- Text input
- Text area

- Radio button
- Checkbox
- Select box
- File select
- Buttons
- Submit and reset buttons
- Hidden input

Table 5 List of form control inputs and their attributes

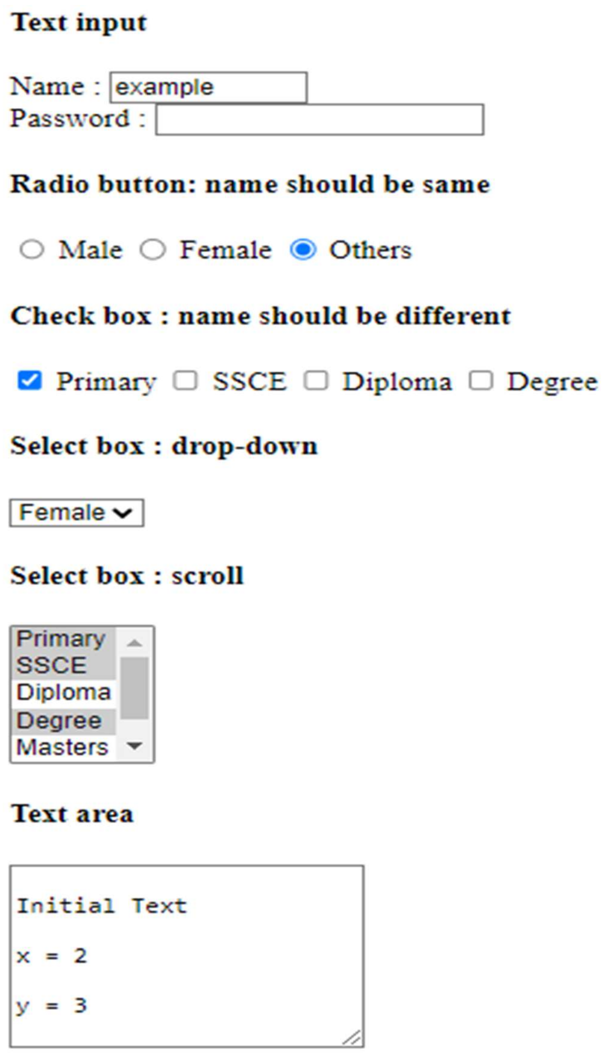
Form Control	**Attributes**	**Values**	**Description**
Input : text	Type	text, password, date, email, number, tel, url, search	different types of inputs
	Value	user-defined	initial value in the area
	Name	user-defined	name send to server
	Size	numeric value	width of the text area
	Maxlength	numeric value	maximum number of characters
Input : radio	Type	Radio	
	Name	user-defined	name send to server
	Value	user-defined value	value of the button if selected
	Checked		check the button by default
Input : check box	Type	Checkbox	
	Name	user-defined	name send to server
	Value	user-defined value	value of the box if selected
	Checked		check the box by default
Input : button	Type	Button	trigger client side script
		Submit	submit the form and run 'action'
		Reset	reset form
		Image	create image button
	Method	get, post	get or post method
	Action	user-defined	action to perform on submit
Input : button	Type	File	create file upload
Input : hidden	Type	Hidden	will not display on html, but can be used
			for sending information to server
Selection box	Name	user-defined	name send to server

	Size	numeric value	enables scroll (default dropdown)
	Multiple	numeric value	select multiple items
	Value	user-defined value	value of the item if selected
	Selected		select item by default
Text area	rows, cols	numeric value	number of rows and cols
	Name	user-defined	name send to server

- Below are the example of the control inputs described in Table 5

```
<!-- Forms -->
      <form>
            <h4>Text input </h4>
            Name : <input type="text" name="user_name" size="10"
            value="example" maxlength="10"><br>
            Password : <input type="password" name="user_pass" ><br>
            <h4> Radio button: name should be same</h4>
            <input type="radio" name="gender"> Male
            <input type="radio" name="gender"> Female
            <input type="radio" name="gender" checked> Others

            <h4> Check box : name should be different</h4>
            <input type="checkbox" name="primary" checked> Primary
            <input type="checkbox" name="ssce"> SSCE
            <input type="checkbox" name="diploma"> Diploma
            <input type="checkbox" name="degree"> Degree

            <h4> Select box : drop-down</h4>
            <select name="s_box">
                  <option value="male">Male</option>
                  <option value="female" selected>Female</option>
                  <option value="others">Others</option>
            </select>

            <h4> Select box : scroll</h4>
            <select name="s_box" size="5" multiple>
                  <option value="primary" selected>Primary</option>
                  <option value="ssce" selected>SSCE</option>
                  <option value="diploma">Diploma</option>
                  <option value="degree" selected>Degree</option>
                  <option value="masters">Masters</option>
                  <option value="phd">PHD</option>
            </select>

            <h4> Text area</h4>
            <textarea rows="10" name="txt_area">
                  Initial Text
                  x = 2
                  y = 3
            </textarea> <!-- formatting work as pre -->
      </form>
```

Code 10 HTML Example 9


Text input

Name : example
Password :

Radio button: name should be same

○ Male ○ Female ● Others

Check box : name should be different

☑ Primary ☐ SSCE ☐ Diploma ☐ Degree

Select box : drop-down

Female

Select box : scroll

Primary
SSCE
Diploma
Degree
Masters

Text area

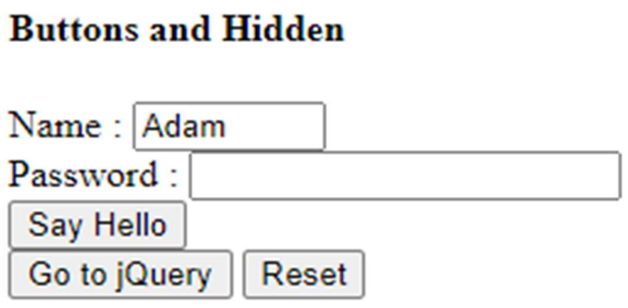
```
Initial Text
x = 2
y = 3
```

Figure 13 HTML Example 9 Output

- Below is the code which shows the working of various buttons. Note that **method** and **action** are defined in this form, which will be triggered on ‘submit’ button. Lastly, ‘hidden’ option is used in this example.

```
<form method="post" action="jquery.html">
      <h4> Buttons and Hidden</h4>
      Name : <input type="text" name="user_name" size="6" value="Adam"
      maxlength="16"><br>
      Password : <input type="password" name="user_pass" ><br>
      <input type="button" onclick="alert('Hello')" name="b_alert" value="Say
      Hello"/><br>
      <input type="submit" name="b_submit" value="Go to jQuery"/>
      <input type="reset" name="b_reset" value="Reset"/><br>
      <input type="hidden" name="h_data" value="html_tutorial">
</form>
```

Code 11 HTML Example 10


Buttons and Hidden

Name : Adam
Password :

Figure 14 HTML Example 10 Output

Table 6 below contains a summary of most of the HTML tags and their brief description.

Table 6 List of HTML Tags and their description

Tag	***Description***
Basic	
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
<br>	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment
Formatting	
<acronym>	Not supported in HTML5. Defines an acronym
<abbr>	Defines an abbreviation
<address>	Defines contact information for the author/owner of a document/article
<b>	Defines bold text
<bdi>New	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<big>	Not supported in HTML5. Defines big text
<blockquote>	Defines a section that is quoted from another source
<center>	Not supported in HTML5. Deprecated in HTML 4.01. Defines centered text
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<del>	Defines text that has been deleted from a document
<dfn>	Defines a definition term
<em>	Defines emphasized text
<font>	Not supported in HTML5. Deprecated in HTML 4.01. Defines font, color, and size for text
<i>	Defines a part of text in an alternate voice or mood

<ins>	Defines a text that has been inserted into a document
<kbd>	Defines keyboard input
<mark>New	Defines marked/highlighted text
<meter>New	Defines a scalar measurement within a known range (a gauge)
<pre>	Defines preformatted text
<progress>New	Represents the progress of a task
<q>	Defines a short quotation
<rp>New	Defines what to show in browsers that do not support ruby annotations
<rt>New	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>New	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<small>	Defines smaller text
<strike>	Not supported in HTML5. Deprecated in HTML 4.01. Defines strikethrough text
<strong>	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<time>New	Defines a date/time
<tt>	Not supported in HTML5. Defines teletype text
<u>	Defines text that should be stylistically different from normal text
<var>	Defines a variable
<wbr>New	Defines a possible line-break
Forms	
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset>, < figure>, or <details> element
<datalist>New	Specifies a list of pre-defined options for input controls
<keygen>New	Defines a key-pair generator field (for forms)
<output>New	Defines the result of a calculation
Frames	
<frame>	Not supported in HTML5. Defines a window (a frame) in a frameset
<frameset>	Not supported in HTML5. Defines a set of frames
<noframes>	Not supported in HTML5. Defines an alternate content for users that do not support frames
<iframe>	Defines an inline frame
Images	
<img>	Defines an image

<map>	Defines a client-side image-map
<area>	Defines an area inside an image-map
<canvas>New	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>New	Defines a caption for a <figure> element
<figure>New	Specifies self-contained content
Audio/Video	
<audio>New	Defines sound content
<source>New	Defines multiple media resources for media elements (<video> and <audio>)
<track>New	Defines text tracks for media elements (<video> and <audio>)
<video>New	Defines a video or movie
Links	
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>New	Defines navigation links
Lists	
<ul>	Defines an unordered list
<ol>	Defines an ordered list
<li>	Defines a list item
<dir>	Not supported in HTML5. Deprecated in HTML 4.01. Defines a directory list
<dl>	Defines a definition list
<dt>	Defines a term (an item) in a definition list
<dd>	Defines a description of an item in a definition list
<menu>	Defines a list/menu of commands
<command>New	Defines a command button that a user can invoke
Tables	
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting
Style/Sections	
<style>	Defines style information for a document
<div>	Defines a section in a document
<span>	Defines a section in a document
<header>New	Defines a header for a document or section
<footer>New	Defines a footer for a document or section
<hgroup>New	Groups heading (<h1> to <h6>) elements

<section>New	Defines a section in a document
<article>New	Defines an article
<aside>New	Defines content aside from the page content
<details>New	Defines additional details that the user can view or hide
<summary>New	Defines a visible heading for a <details> element
Meta Info	
<head>	Defines information about the document
<meta>	Defines metadata about an HTML document
<base>	Specifies the base URL/target for all relative URLs in a document
<basefont>	Not supported in HTML5. Deprecated in HTML 4.01. Specifies a default color, size, and font for all text in a document
Programming	
<script>	Defines a client-side script
<noscript>	Defines an alternate content for users that do not support client-side scripts
<applet>	Not supported in HTML5. Deprecated in HTML 4.01. Defines an embedded applet
<embed>New	Defines a container for an external (non-HTML) application
<object>	Defines an embedded object
<param>	Defines a parameter for an object

(b) CSS

(i) What is CSS?

Cascading style sheets (CSS) is a W3C standard for describing the appearance of HTML elements. Another common way to describe CSS's function is to say that CSS is used to define the **presentation** of HTML documents. With CSS, we can assign font properties, colors, sizes, borders, background images, and even position elements on the page. CSS can be added directly to any HTML element (via the `style` attribute), within the `<head>` element, or, most commonly, in a separate text file that contains only CSS.

(ii) Benefits of CSS

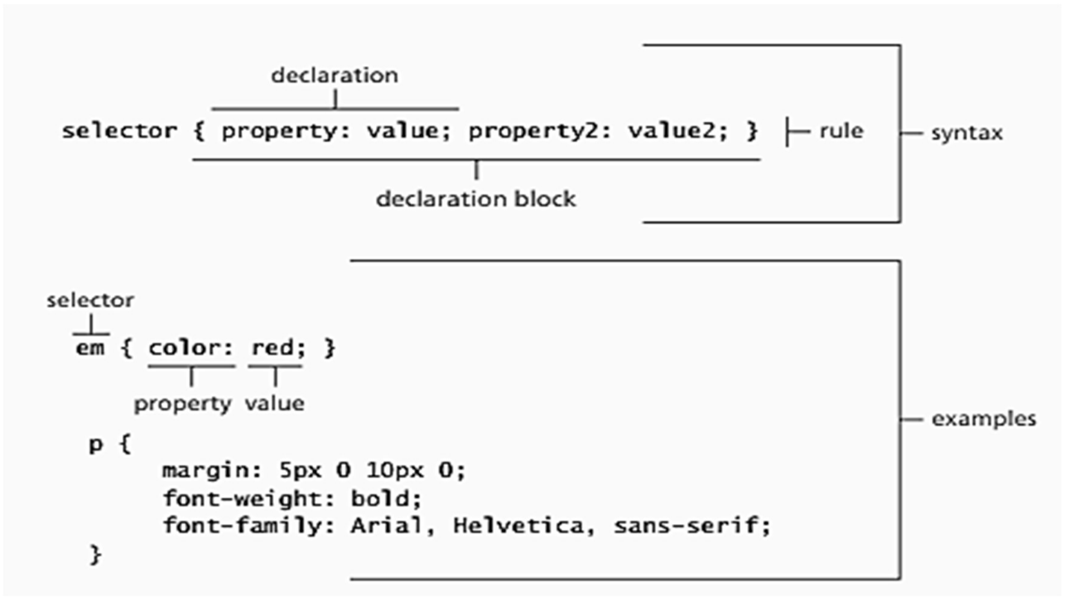
Before digging into the syntax of CSS, we should see a few reasons why using CSS is a better way of describing appearances than HTML alone. The benefits of CSS include:

- **Improved control over formatting.** The degree of formatting control in CSS is significantly better than that provided in HTML. CSS gives web authors fine-grained control over the appearance of their web content.
- **Improved site maintainability.** Websites become significantly more maintainable because all formatting can be centralized into one CSS file, or a small handful of them. This allows you to make site-wide visual modifications by changing a single file.

- **Improved accessibility.** CSS-driven sites are more accessible. By keeping presentation out of the HTML, screen readers and other accessibility tools work better, thereby providing a significantly enriched experience for those reliant on accessibility tools.
- **Improved page download speed.** A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain less style information and markup, and thus be smaller.
- **Improved output flexibility.** CSS can be used to adopt a page for different output media. This approach to CSS page design is often referred to as responsive design. Figure 3 illustrates a site that responds to different browser and window sizes.

(iii) CSS syntax

A CSS document consists of one or more style rules. A rule consists of a **selector** that identifies the HTML element or elements that will be affected, followed by a series of **property: value** pairs (each pair is also called a **declaration**), as shown in Figure 15. The series of declarations is also called the **declaration block**. As one can see in the illustration, a declaration block can be together on a single line, or spread across multiple lines. The browser ignores white space (i.e., spaces, tabs, and returns) between your CSS rules so you can format the CSS however you want. Notice that each declaration is terminated with a semicolon. The semicolon for the last declaration in a block is in fact optional. However, it is sensible practice to also terminate the last declaration with a semicolon as well; that way, if you add rules to the end later, you will reduce the chance of introducing a rather subtle and hard-to-discover bug.



```
selector { property: value; property2: value2; }

em { color: red; }

p {
    margin: 5px 0 10px 0;
    font-weight: bold;
    font-family: Arial, Helvetica, sans-serif;
}
```

Figure 15 CSS Syntax

(iv) CSS Selectors

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern that is used by the browser to select the HTML elements that will receive the style. When defining CSS rules, you will need to first use a selector to tell the browser which elements will be affected by the property values. CSS selectors allow you to select individual or multiple HTML elements. There are a variety of ways to write selectors, but we'll look at the three basic selector types:

1. **Element Selectors:**

   Element selectors select all instances of a given HTML element. The example CSS rules in Figure 15 illustrate two element selectors. You can select all elements by using the universal element selector, which is the * (asterisk) character. You can select a group of elements by separating the different element names with commas. This is a sensible way to reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule. Below is an example of grouped selector along with its equivalent as three separate rules.

```
/* commas allow you to group selectors */
body, p, div {
margin: 0;
padding: 0;
}
/* the above single grouped selector is equivalent to the
following: */
body {
margin: 0;
padding: 0;
}
p {
margin: 0;
padding: 0;
}
div {
margin: 0;
padding: 0;
}
```

2. **Class Selectors:**

   A class selector allows you to simultaneously target different HTML elements regardless of their position in the document tree. If a series of HTML elements have been labeled with the same class attribute value, then you can target them for styling by using a class selector, which takes the form: period (.) followed by the class name. Below is an example of styling using a class selector.

```
<html>
     <head>
          <title>My Web Page</title>
          <style>
               .heading {
                    font-style: Bold;
                    color: red;
               }
          </style>
     </head>
     <body>
          <h1 class="heading">About Me</h1>
          <div>
               <p>Welcome to my Web page</p>
          </div>
     </body>
</html>
```

3. **Id Selectors:**

An id selector allows you to target a specific element by its id attribute regardless of its type or position. If an HTML element has been labeled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name. An example of styling using an id selector is shown below.

```
<html>
     <head>
          <title>My Web Page</title>
          <style>
               #greetings {
                    font-style: italic;
                    color: red;
               }
          </style>
     </head>
     <body>
          <h1>Home</h1>
          <div id="greetings">
               <p>By Adam on <time>January 30,
               2023</time></p>
               <p>Welcome to my Web page.</p>
          </div>
     </body>
</html>
```

(v) Properties

Each individual CSS declaration must contain a property. These property names are predefined by the CSS standard. The CSS2.1 recommendation defines over a hundred different property names, so some type of reference guide, whether in a book, online, or within your web development software, can be helpful. Below is a list of the most commonly used CSS properties.

Property-type	**Property**
Fonts	font
	font-family
	font-size
	font-style
	font-weight
	@font-face
Text	letter-spacing
	line-height
	text-align
	text-decoration
	text-indent
Color and background	background
	background-color
	background-image
	background-position
	background-repeat
	color
Borders	border
	border-color
	border-width
	border-style
	border-top
	border-top-color
	border-top-width
	etc.

(vi) Values

Each CSS declaration also contains a value for a property. The unit of any given value is dependent upon the property. Some property values are from a predefined list of keywords. Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

CSS supports a variety of different ways of describing color; Below is a list of the different ways you can describe a color value in CSS.

Method	**Description**	**Example**
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	color: red; color: hotpink; /*CSS3 only */

RGB	Uses three different numbers between 0 and 255 to describe the red, green, and blue values of the color.	color: rgb(255,0,0);
		color: rgb(255,105,180);

Hexadecimal	Uses a six-digit hexadecimal number to describe the red, green, and blue value of the color; each of the three RGB values is between 0 and FF (which is 255 in decimal). Notice that the hexadecimal number is preceded by a hash or pound symbol (#).	color: #FF0000;
		color: #FF69B4;

RGBa	This defines a partially transparent background color. The “a” stands for “alpha”, which is a term used to identify a transparency that is a value between 0.0 (fully transparent) and 1.0 (fully opaque).	color: rgb(255,0,0, 0.5);
HSL	Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well.	color: hsl(0,100%,100%); color: hsl(330,59%,100%)

Just as there are multiple ways of specifying color in CSS, so too there are multiple ways of specifying a unit of measurement. These units can sometimes be complicated to work with. When working with print design, we generally make use of straightforward absolute units such as inches or centimeters and picas or points. However, because different devices have differing physical sizes as well as different pixel resolutions and because the user is able to change the browser size or its zoom mode, these absolute units don’t always make sense with web element measures.

(vii) Location of styles

As mentioned earlier, CSS style rules can be located in three different locations. These three are not mutually exclusive, in that you could place your style rules in all three. In practice, however, web authors tend to place all of their style definitions in one (or more) external style sheet files. These files are created using text editors (like Notepad) and saved with the **.css** file extension.

1. **Inline Styles**

   Inline styles are style rules placed within an HTML element via the style attribute, as shown in below. An inline style only affects the element it is defined within and overrides any other style definitions for properties used in the inline style. Notice that a selector is not necessary with inline styles and that semicolons are only required for separating multiple rules.

   ```
   <h1>About Myself</h1>
   <h2style="font-size: 24pt">Biography</h2>
   ...
   <h2 style="font-size: 24pt; font-weight: bold;">Education</h2>
   ```

   Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability (because presentation and content are intermixed and because it can be difficult to make consistent inline style changes across multiple files). Inline styles can, however, be handy for quickly testing out a style change.

2. **Embedded Style Sheet**

   Embedded style sheets (also called internal styles) are style rules placed within the `<style>` element (inside the `<head>` element of an HTML document), as shown below. While better than inline styles, using embedded styles is also by and large discouraged. Since each HTML document has its own `<style>` element, it is more difficult to consistently style multiple documents when using embedded styles. Just as with inline styles, embedded styles can, however, be helpful when quickly testing out a style that is used in multiple places within a single HTML document.

   ```
   <html>
         <head>
               <title>My Web Page</title>
               <style>
                     h1 {
                           font-size: 24pt;
                     }
   ```

```
                h2 {
                     font-size: 18pt;
                     font-weight: bold;
                }
           </style>
      </head>
      <body>
           <h1>Welcome</h1>
           <h2>About Me</h2>
           ...
      </body
</html>
```

3. **External Style Sheet**

   External style sheets are style rules placed within an external text file with the .css extension. This is by far the most common place to locate style rules because it provides the best maintainability. When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version. The browser is able to cache the external style sheet, which can improve the performance of the site as well. To reference an external style sheet, you must use a `<link>` element (within the `<head>` element), as shown below. You can link to several style sheets at a time; each linked style sheet will require its own `<link>` element.

```
<html>
     <head>
          <title>My Web Page</title>
               <link rel="stylesheet" href="styles.css">
     </head>
     <body>
          <h1>Greetings</h1>
          <p> Welcome to my Web page.</p>
     </body>
</html>
```

(viii) The Universal Selector

The * **wildcard** or **universal selector** matches any element, so the following rule will make a complete mess of a document by giving a green border to all of its elements:

```
* { border:1px solid green; }
```

It's therefore unlikely that you will use the * on its own, but as part of a compound rule it can be very powerful. For example, the following rule will apply the same styling as the preceding one, but only to all paragraphs that are sub-elements of the element with the ID boxout, and only as long as they are not direct children:

```
#boxout * p {border:1px solid green; }
```

Let’s look at what’s going on here. The first selector following *#boxout* is a * symbol, so it refers to any element within the boxout object. The following p selector then narrows down the selection focus by changing the selector to apply only to paragraphs (as defined by the p) that are sub-elements of elements returned by the * selector. Therefore, this CSS rule performs the following actions:

a) Find the object with the ID of boxout.
b) Find all sub-elements of the object returned in step 1.
c) Find all p sub-elements of the objects returned in step 2 and, because this is the final selector in the group, also find all p sub- and sub-sub-elements (and so on) of the objects returned in step 2.
d) Apply the styles within the {and } characters to the objects returned in step 3.

The net result of this is that the green border is applied only to paragraphs that are grandchildren (or great-grandchildren, etc.) of the main element.

(ix) Using Comments

It is a good idea to comment your CSS rules, even if you describe only the main groups of statements rather than all or most of them. You can do this in two different ways. First, you can place a comment within a pair of /* ... */ tags, like this:

```
/* This is a CSS comment */
```

Or you can extend a comment over many lines, like this:

```
/*

A Multi

line

comment

*/
```

(x) Measurements

CSS supports an impressive range of units of measurement, enabling you to tailor your web pages precisely to specific values, or by relative dimensions. The ones generally used are **pixels**, **points**, **ems**, and **percent**, but here's the complete list:

1) Pixels

The size of a pixel varies according to the dimensions and pixel depth of the user's monitor. One pixel equals the width/height of a single dot on the screen, and so this measurement is best suited to monitors. For example:

```
.classname { margin:5px; }
```

2) Points

A point is equivalent in size to 1/72 of an inch. The measurement comes from a print design background and is best suited for that medium, but is also commonly used on monitors. For example:

```
.classname { font-size:14pt; }
```

3) Inches

An inch is the equivalent of 72 points and is also a measurement type best suited for print. For example:

```
.classname { width:3in; }
```

4) Centimeters

Centimetres are another unit of measurement best suited for print. One centimetre is a little over 28 points. For example:

```
.classname { height:2cm; }
```

5) Millimetres

A millimetre is 1/10 of a centimetre (or almost 3 points). Millimetres are another measure best suited to print. For example:

```
.classname { font-size:5mm; }
```

6) Picas

A pica is another print typographic measurement, which is equivalent to 12 points. For example:

```
.classname { font-size:1pc; }
```

7) Ems

An em is equal to the current font size and is therefore one of the more useful measurements for CSS because it is used to describe relative dimensions. For example:

```
.classname { font-size:2em; }
```

8) Exs

An ex is also related to the current font size; it is equivalent to the height of a lowercase letter x. This is a less popular unit of measurement that is most often used as a good approximation for helping to set the width of a box that will contain some text. For example:

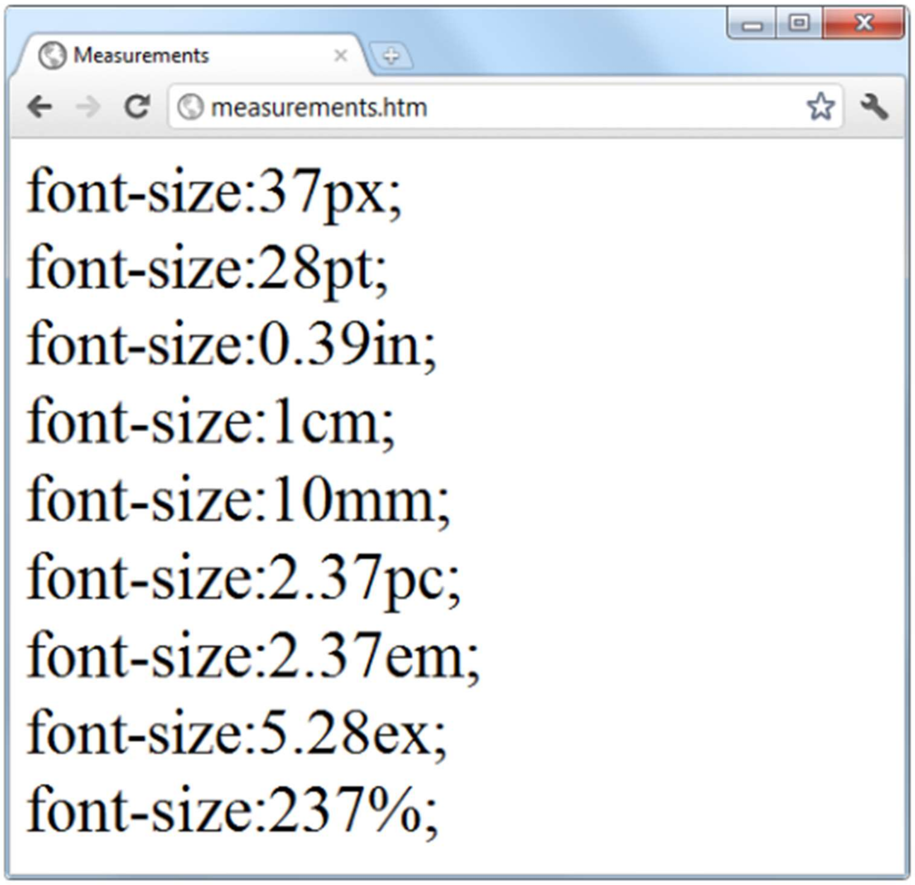
```
.classname { width:20ex; }
```

9) Percent

This unit is related to the em in that it is exactly 100 times greater (when used on a font). Whereas 1 em equals the current font size, the same size is 100 in percent. When not relating to a font, this unit is relative to the size of the container of the property being accessed. For example:

```
.classname { height:120%; }
```

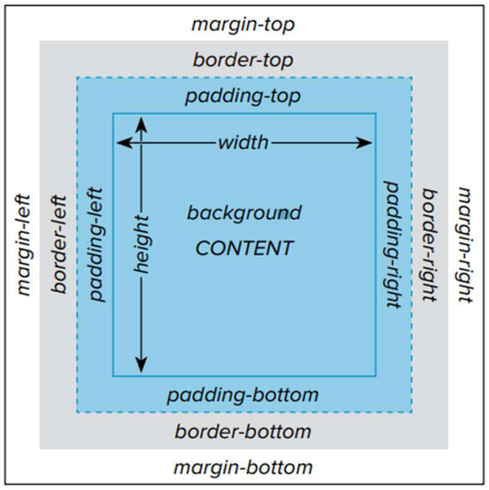
Figure 15 shows each of these measurement types in turn being used to display text in almost identical sizes.



font-size:37px;
font-size:28pt;
font-size:0.39in;
font-size:1cm;
font-size:10mm;
font-size:2.37pc;
font-size:2.37em;
font-size:5.28ex;
font-size:237%;

Figure 16 Different measurements that display almost the same font size

(xi) The Box Model and Layout



The CSS properties affecting the layout of a page are based around the box model, a nested set of properties surrounding an element. Virtually all elements have (or can have) these properties, including the document body, whose margin you can, for example, remove with the following rule:

```
body { margin:0px; }
```

The box model of an object starts at the outside, with the object's **margin**. Inside this is the **border**, then there is **padding** between the border and the inner contents, and finally there's the object's **contents**. Once you have the hang of the box model, you will be well on your way to creating professionally laid-out pages, as these properties alone will make up much of your page styling.

1) Setting Margins

The margin is the outermost level of the box model. It separates elements from each other and its use is quite smart. For example, assume you have chosen to give a number of elements a default margin of 10 pixels around each. When they are placed on top of each other, this would create a gap of 20 pixels (the total of the adjacent border widths).

CSS overcomes this potential issue, however: when two elements with borders are positioned directly one above the other, only the larger of the two margins is used to separate them. If both margins are the same width, just one of the widths is used. This way, you are much more likely to get the result you want. But you should note that the margins of absolutely positioned or inline elements do not collapse.

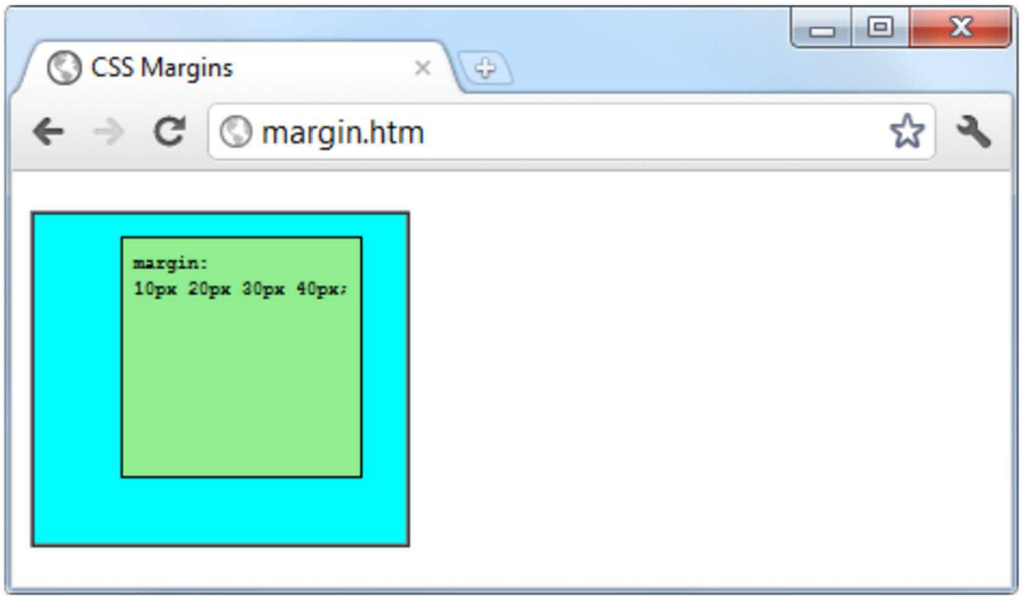
The margins of an element can be changed en masse with the margin property, or individually with margin-left, margin-top, margin-right, and margin-bottom. When setting the margin property, you can supply one, two, three, or four arguments, which have the effects commented in the following rules:

```
/* Set all margins to 1 pixel */
margin:1px;
/* Set top and bottom to 1 pixel, and left and right to 2 */
margin:1px 2px;
/* Set top to 1 pixel, left and right to 2, and bottom to 3 */
margin:1px 2px 3px;
/* Set top to 1 pixel, right to 2, bottom to 3, and left to 4
*/
margin:1px 2px 3px 4px;
```

Figure 16 shows Code example 12 loaded into a browser, with the margin property rule (highlighted in bold) applied to a square element that has been placed inside a table element. The table has been given no dimensions, so it will simply wrap as closely around the inner <div> element as it can. As a consequence, there is a margin of 10 pixels above it, 20 pixels to its right, 30 pixels below it, and 40 pixels to its left.

```
<!DOCTYPE html>
<html>
      <head>
            <title>Margins</title>
            <style>
                  #object1 {
                        background :lightgreen;
                        border-style:solid;
                        border-width:1px;
                        font-family :"Courier New";
                        font-size :9px;
                        width :100px;
                        height :100px;
                        padding :5px;
                        margin :10px 20px 30px 40px;
                  }
                  table {
                        padding :0;
                        border :1px solid black;
                        background :cyan;
                  }
            </style>
      </head>
      <body>
            <table>
                  <tr>
                        <td>
                              <div id='object1'>margin:<br>10px
                              20px 30px 40px;</div>
                        </td>
                  </tr>
            </table>
      </body>
</html>
```

Code 12 How margins are applied



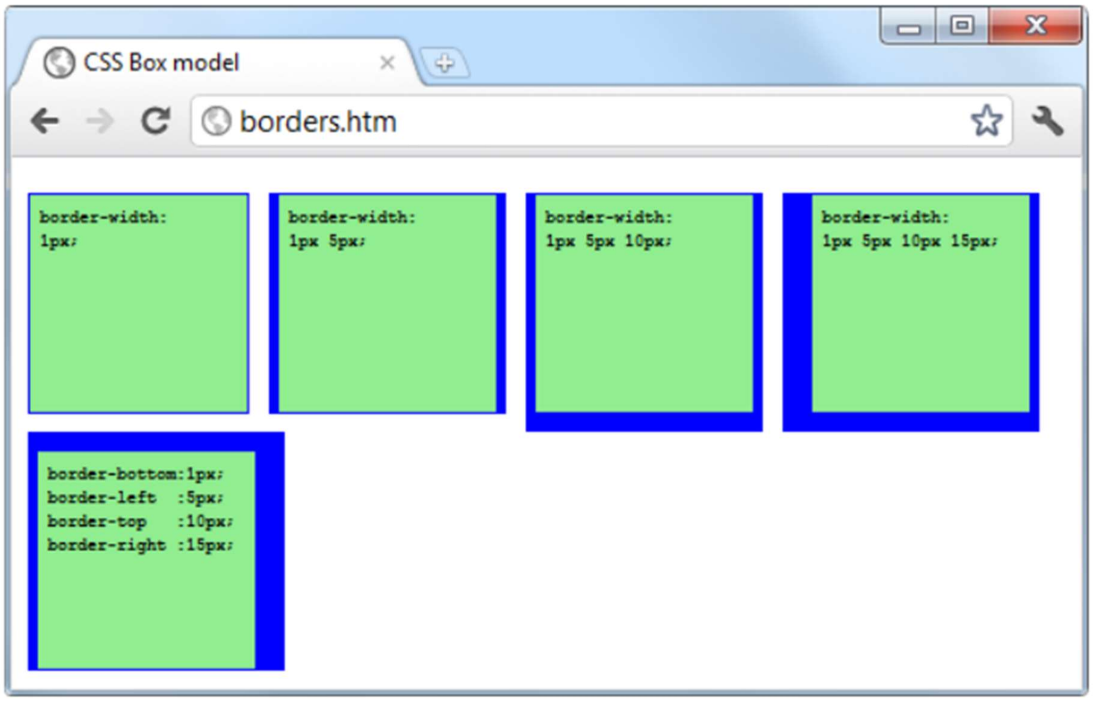
```
margin:
10px 20px 30px 40px;
```

Figure 17 The outer table expands according to the margin widths

2) Applying Borders

The border level of the box model is similar to the margin except that there is no collapsing. It is the next level as we move into the box model. The main properties used to modify borders are border, border-left, border-top, border-right, and borderbottom, and each of these can have other subproperties added as suffixes, such as -color, -style, and -width. The four ways to access individual property settings used for the margin property also apply with the border-width property, so all the following are valid rules:

```
/* All borders */
border-width:1px;
/* Top/bottom left/right */
border-width:1px 5px;
/* Top left/right bottom */
border-width:1px 5px 10px;
/* Top right bottom left */
border-width:1px 5px 10px 15px;
```



```
border-width:
1px;

border-width:
1px 5px;

border-width:
1px 5px 10px;

border-width:
1px 5px 10px 15px;

border-bottom:1px;
border-left  :5px;
border-top   :10px;
border-right :15px;
```

Figure 18 Applying long- and shorthand border rule values

Figure 18 shows each of these rules applied in turn to a group of square elements. In the first one, you can clearly see that all borders have a width of 1 pixel. The second element, however, has a top and bottom border width of 1 pixel, while its side widths are 5 pixels each.

The third element has a 1 pixel wide top; its sides are 5 pixels wide, and its bottom is 10 pixels wide. The fourth element has a 1-pixel top border width, a 5-pixel right border width, a 10-pixel bottom border width, and a 15-pixel left border width.

The final element, under the previous ones, doesn't use the shorthand rules; instead, it has each of the border widths set separately. As you can see, it takes a lot more typing to achieve the same result.

3) Adjusting Padding

The deepest of the box model levels (other than the contents of an element) is the padding, which is applied inside any borders and/or margins. The main properties used to modify padding are padding, padding-left, padding-top, padding-right, and padding-bottom.

The four ways of accessing individual property settings used for the margin and border properties also apply with the padding property, so all the following are valid rules:

```
/* All padding */
padding:1px;
/* Top/bottom and left/right */
padding:1px 2px;
/* Top, left/right and bottom */
padding:1px 2px 3px;
/* Top, right, bottom and left */
padding:1px 2px 3px 4px;
```

(c) JavaScript

JavaScript brings a dynamic functionality to your websites. Every time you see something pop up when you mouse over an item in the browser, or see new text, colors, or images appear on the page in front of your eyes, or grab an object on the page and drag it to a new location – all those things are done through JavaScript. It offers effects that are not otherwise possible, because it runs inside the browser and has direct access to all the elements in a web document.

JavaScript first appeared in the Netscape Navigator browser in 1995, coinciding with the addition of support for Java technology in the browser. Because of the initial incorrect impression that JavaScript was a spin-off of Java, there has been some long-term confusion over their relationship. However, the naming was just a marketing ploy to help the new scripting language benefit from the popularity of the Java programming language.

JavaScript gained new power when the HTML elements of the web page got a more formal, structured definition in what is called the **Document Object Model**, or **DOM**. The DOM makes it relatively easy to add a new paragraph or focus on a piece of text and change it.

Because both JavaScript and PHP support much of the structured programming syntax used by the C programming language, they look very similar to each other. They are both fairly high-level languages, too; for instance, they are weakly typed, so it's easy to change a variable to a new type just by using it in a new context.

JavaScript is at the heart of the Web 2.0 Ajax technology that provides the fluid web frontends that (along with HTML5 features) savvy web users expect these days.

Here are some points to note about JavaScript:

- JavaScript is a dynamic language which is used for designing the web pages on the client side
- It is case sensitive language.
- It is untyped language i.e. a variable can hold any type of value.
- // is used for comments.
- ; i used for line termination.
- JavaScript code should be added at the end i.e. just above the closing-body-tag.
- It is better to write the JavaScript code in separate file.

(i) Basics of JavaScript

JavaScript is a vast programming language that covers several concepts and most importantly, many of these concepts in different ways. While learning JavaScript in this course, it is necessary to go through some of the basics of JavaScript programming language.

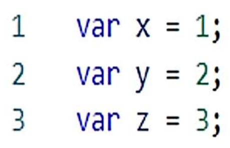
1) Variables

Let's start with one of the most basic terms used in programming - variable. A variable is defined as **a container that is used to hold values.**

```
var x;
var y;
var z;
```

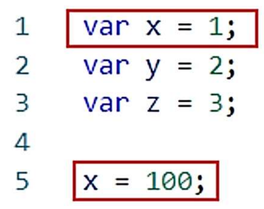
x, y, and z are three variables that are **declared using the “var” keyword.** Two other keywords can also be used to declare variables - **let and const.**

As of now, these variables do not hold any values. To initialize them, **We use the assignment operator (=).**



```
var x = 1;
var y = 2;
var z = 3;
```

Declaring a variable with values is called initializing. We can also change these values later by using the assignment operator.



```
var x = 1;
var y = 2;
var z = 3;

x = 100;
```

2) Data types

A variable holds data. Several types of data can be stored in a variable. In the above example, All the variables were holding numerical values. We can also store **single-precision numbers, double-precision numbers, strings, boolean values, objects, date,** etc, in a variable. These are different data types used in the programming world. A data type indicates the characteristics of the data stored in a variable.

In the other programming languages such as Java, it is mandatory to specify the data type of a variable while declaring, and moreover, a variable can hold only that type of data. But there are no such restrictions in JavaScript.

Data types in JavaScript are divided into two categories **Primitive and Non-primitive.**

Following is the list of Primitive data types:

- Number
- String
- Boolean
- Undefined

```
var num = 100;
var str = "Hello World";
var bool = true;
var und = undefined;
```

Following is the list of non-primitive data types:

- Object
- Array
- Date

```
var obj = {
    str : "Hello World"
}

var date = new Date();

var arr = [1,2,3,4,5]
```

If you have knowledge of any popular programming language, you will easily understand all these data types.

3) Functions

While coding, we may face situations when we need to execute the same code at different times. For example, we need to find the sum of two numbers five times in a program. The numbers are different every time. we need to write the same logic for five times throughout the program. It will work fine, but do we think it is efficient? The answer is no. It is not efficient because writing the same part again and again, not only increases the length of the code but also the time and effort. Here enters the concept of functions.

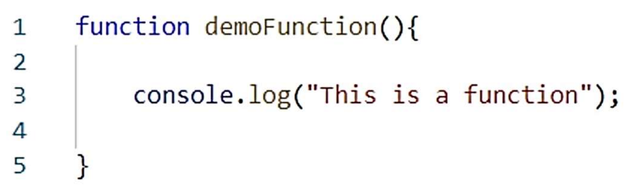
Functions are called the main building blocks of a program. They provide **code reusability and helps reducing time and effort.**

For the above example, we can create a function, that accepts two values and returns their sum. we can call this function anytime anywhere we want. Thus, removing the need for repetition.

Functions are one of the most important parts of programming. Almost every language has the concept of functions. In this chapter, we will discuss all about functions in JavaScript.

a) Declaring a function

To create a function in JavaScript, we need **the 'function' keyword**. Observe the following JavaScript code.



```
function demoFunction(){

    console.log("This is a function");

}
```

A function in JavaScript is declared using the 'function' keyword followed by the name of the function and two parentheses. In the above code, the name of the function is 'demoFunction'. we can also see parenthesis attached to it.

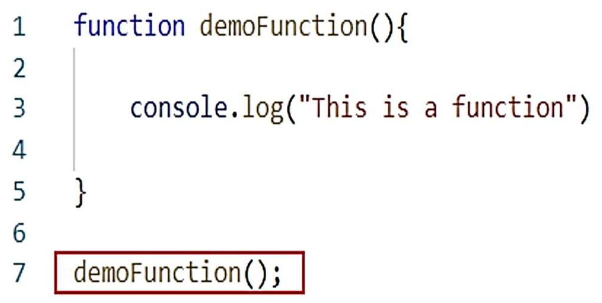
These parentheses can be empty or can have parameters. We will discuss the parameters later. Currently, there are no parameters for this function.

The body of the function is written inside the curly brackets, similar to if and switch statements.

b) Calling a function

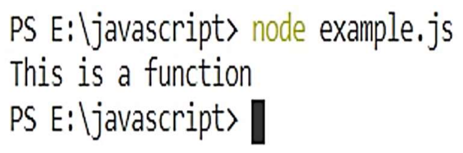
Declaring a function is not enough. **Nothing will happen until the function is called**.

To call a function, simply write the name of the function with the parenthesis.



```
function demoFunction(){

    console.log("This is a function")

}

demoFunction();
```

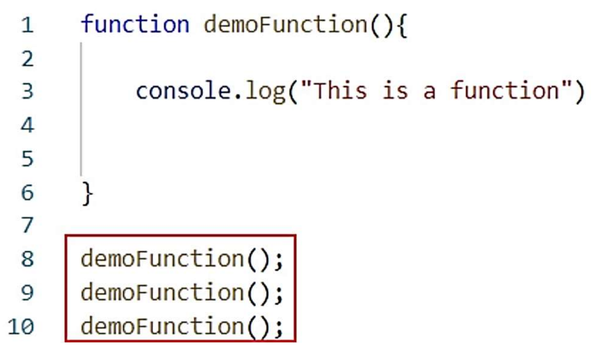
Let's execute the above code and check the output.



```
PS E:\javascript> node example.js
This is a function
PS E:\javascript> 
```

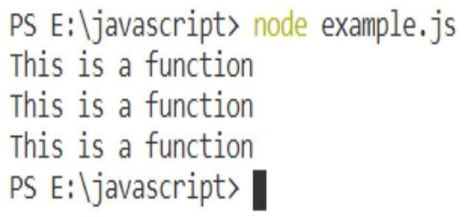
The function is executed once because I called it only a single time.

The main objective of using functions is to avoid code repetition and offer better code reusability. Let me show how this is possible with functions.



```
function demoFunction(){

    console.log("This is a function")


}

demoFunction();
demoFunction();
demoFunction();
```

The demoFunction() is called three times.

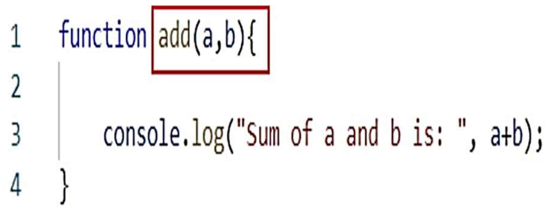


```
PS E:\javascript> node example.js
This is a function
This is a function
This is a function
PS E:\javascript>
```

The console statement inside the demoFunction() is executed three times while actually, it was written once. This is the benefit of using functions. Create a function and call it anytime and anywhere we want.

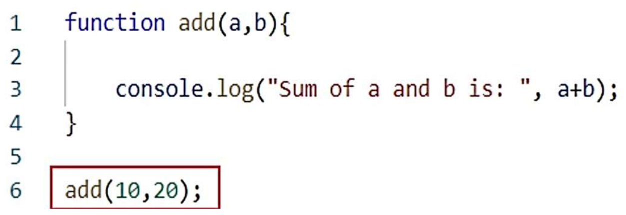
c) Parameters

we can always pass data to a function. **The data is passed as parameters** or also known as arguments. Let's go to the example we were discussing earlier in this lesson.



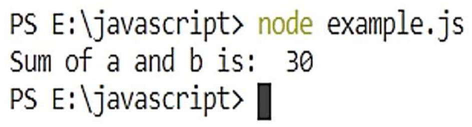
```
function add(a,b){

    console.log("Sum of a and b is: ", a+b);
}
```

Observe the add() function. we can see, it has two parameters a and b, which are then used inside the function body. Similarly, we need to pass the values of these parameters while calling the function.



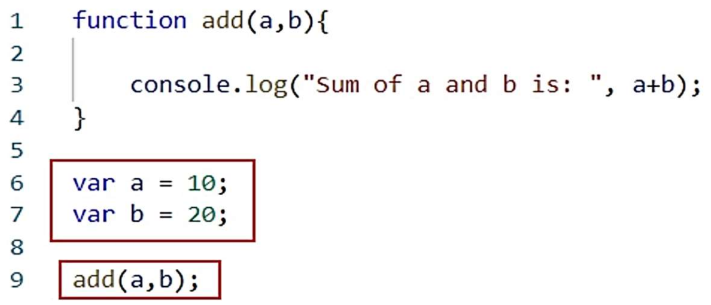
```
function add(a,b){

    console.log("Sum of a and b is: ", a+b);
}

add(10,20);
```

The value of a is 10 and b is 20. Let’s check the output.



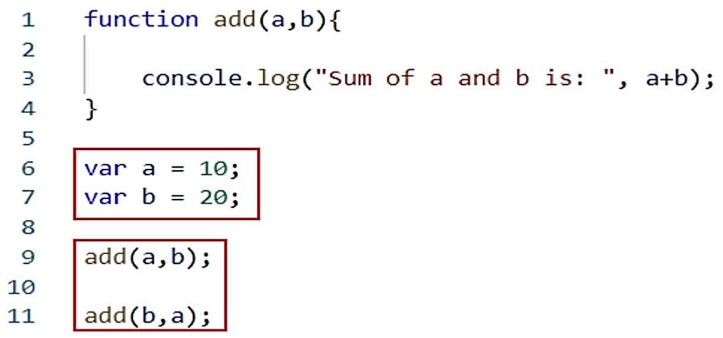
```
PS E:\javascript> node example.js
Sum of a and b is:  30
PS E:\javascript> 
```

we can also pass the variables to a function as parameters.



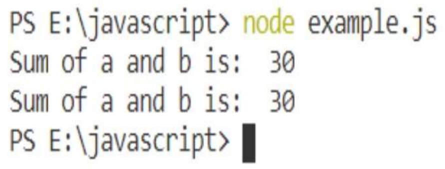
```
function add(a,b){

    console.log("Sum of a and b is: ", a+b);
}

var a = 10;
var b = 20;

add(a,b);
```

But remember, **the names of parameters in the function declaration have no relation with the name of the variables passed as the parameters while function calling.**



```
function add(a,b){

    console.log("Sum of a and b is: ", a+b);
}

var a = 10;
var b = 20;

add(a,b);

add(b,a);
```

- During the first function call - add(a,b), the value of a in the add() function is 10 and b is 20.
- During the second function call - add(b,a), the value of a in the add() function is 20 and b is 10.



```
PS E:\javascript> node example.js
Sum of a and b is:  30
Sum of a and b is:  30
PS E:\javascript> 
```

Summary

- A variable can be declared using var, let, or const keywords.
- To assign a value to a variable, use the assignment operator.
- There are two categories of data types in JavaScript Primitive and Non-primitive.

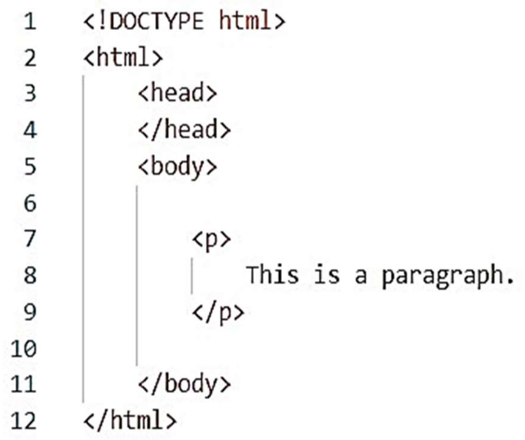
- The primitive data types consist of Number, String, Boolean, and Undefined.
- The Non-primitive data types consist of Object, Date, and Array.
- Functions are the code of blocks that are used for code reusability.
- A function is created using the function keyword and it can be called any number of times using the function name.
- The values passed to a function are called parameters.

4) DOM

DOM stands for **Document Object Model** . It is a very important concept on the road to becoming a web developer. So in this chapter, we will discuss what is DOM and how it is used with JavaScript.

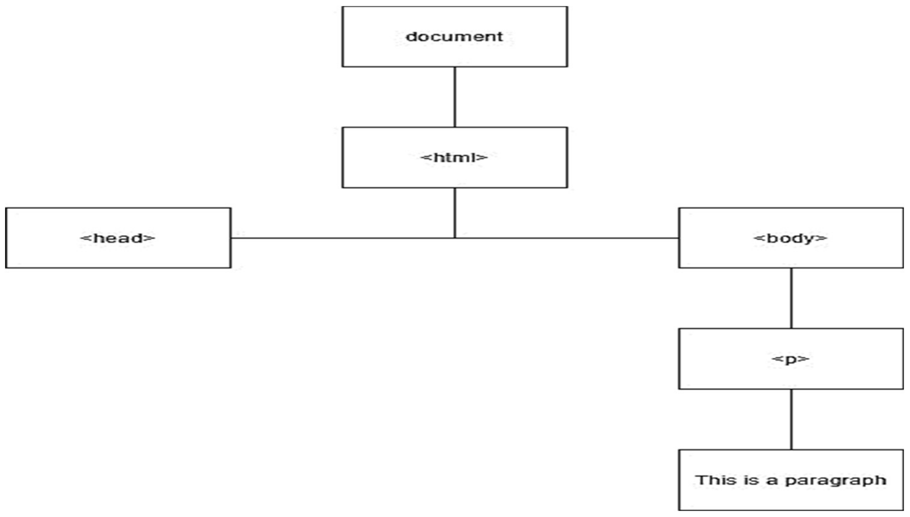
a) What is DOM?

When the HTML file is loaded into a browser, **a tree-like structure** is created. This structure has **various nodes** , and these **nodes represent various elements of the HTML document.**



```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p>
            This is a paragraph.
        </p>

    </body>
</html>
```

The DOM structure of the HTML document will look like the following structure.



The structure starts from <html> tag, that is also the root element. Then, <head> and <body> are two different nodes. In the body, there is a <p> tag which is a node of the <body> tag.

The tree structure will grow as the more tags are added in the HTML document.

b) Use of DOM

So far, we created HTML documents using various tags and then applied CSS. Everything is still static. Once loaded in the browser, nothing changes. But websites are not like that.

Suppose, there is a button and when clicked on that button, something happens. Say, the button click changes the text of a <p> tag. This is done by **manipulating DOM with the help of JavaScript.** In simple words, JavaScript is used to create dynamic HTML by making changes in the DOM tree.

Take the above example. To change the content of the <p> tag, we will use JavaScript to access that node, and then by using various DOM methods, it is manipulated accordingly.

With JavaScript, we can manipulate almost everything in HTML such as content, styles, even add new elements and remove the existing ones.

Summary

- DOM is a tree-like structure that is created when an HTML document is loaded in a browser.
- Every node of a DOM tree represents an HTML element.
- DOM can be manipulated to make dynamic changes in an HTML document.

5) HTML events and JavaScript

a) HTML events

HTML events are **attributes** that are used to make something happen. For example, a button click popping a message. Another example is, popping a message when page loads or when the input changes.

HTML events are very important because they are used to **convert the static HTML elements into dynamic.** One of the most important uses of these events is that JavaScript functions can be triggered using them. Although a very basic DOM manipulation can be done using HTML, serious manipulation is done using functions. This is why HTML events are very important.

There are several HTML events. They are divided into different categories:

- Keyboard events

- Mouse events
- Drag events
- Form events
- Windows events
- Media events
- Clipboard events

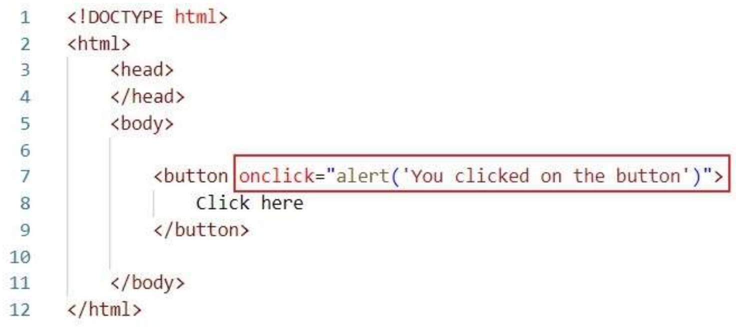
Further, there are several events in each of these categories. The main focus of studying the events is to understand how to use these events to trigger some action like those mentioned below:.

1. Triggering alert()

   The **alert()** function is pop up that appears on the screen with a message. Let's see how can we trigger a pop up using the onclick mouse event.

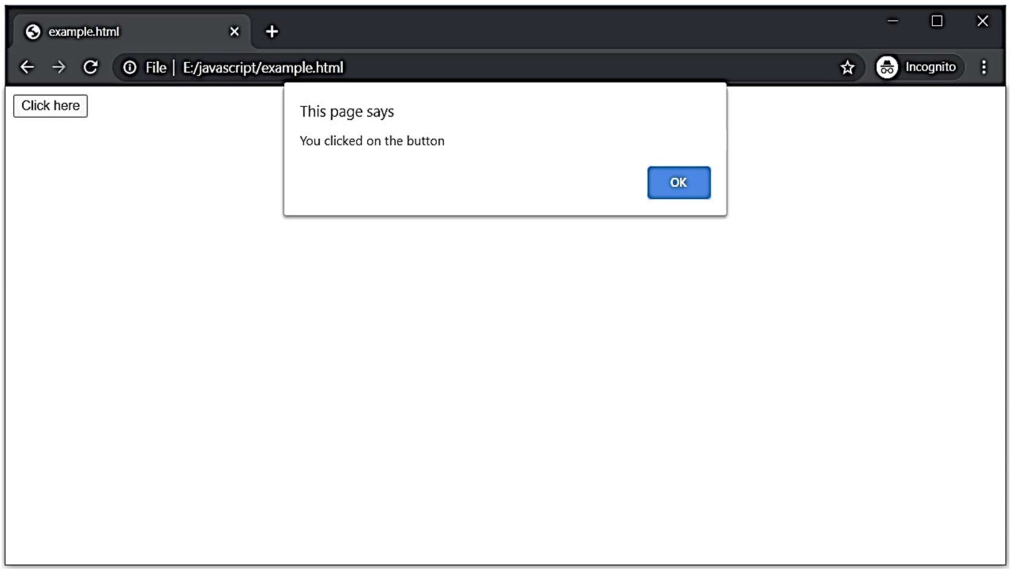
   Mouse events are one of the most commonly used HTML events. These events are triggered when the user does something with the mouse. For example, clicking on something, double-clicking on something, hovering over something, and many more.

   The onclick event is the most basic HTML event. As the name suggests, this **event triggers something when an element is clicked on.**



```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="alert('You clicked on the button')">
            Click here
        </button>

    </body>
</html>
```

As mentioned earlier, HTML events are attributes. So the button above has an onclick event and its value is an alert() function. Remember, **the value of an HTML event is always written inside quotes.** The value will be triggered when the button is clicked.



This page says

You clicked on the button

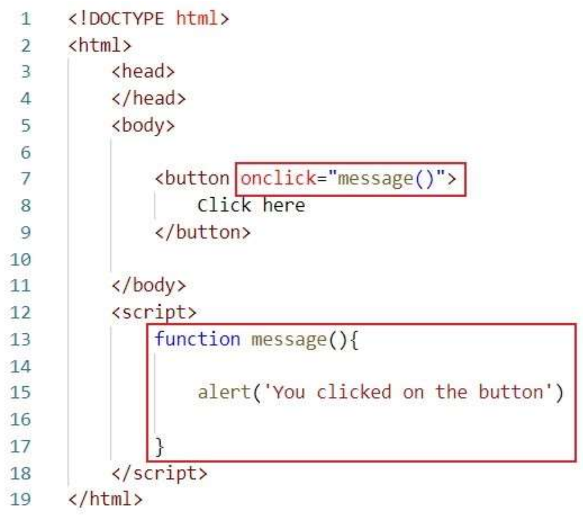
The pop up appears when the button is clicked. So this example was to give you a basic understanding of how HTML events are used.

2. Triggering a function

   In the real-time, events are used to trigger JavaScript functions.

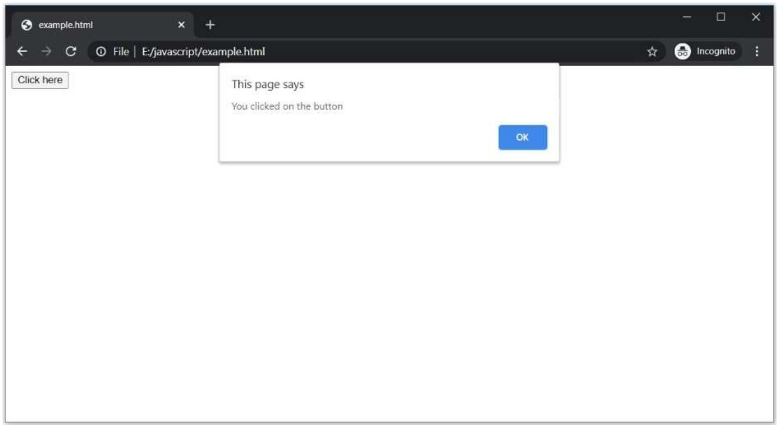
   Generally, complicated things such as DOM manipulation require some kind of coding logic and there can be multiple lines of code. So it is not possible to write all these lines in the HTML, and that is why HTML functions are used.

   Let's put the alert() into a function.



```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="message()">
            Click here
        </button>

    </body>
    <script>
        function message(){

            alert('You clicked on the button')

        }
    </script>
</html>
```

The function is placed inside the <script> tag, in fact, all the JavaScript is placed inside this tag only or in a separate JavaScript file. To trigger this function, we have to give it as the value of the onclick event.



This page says
You clicked on the button

This is how we trigger a function using HTML events.

Commonly used HTML events

Let’s discuss some of the commonly used HTML events categorically.

1. Mouse events
   - onclick - triggers on the single click of the mouse.
   - ondblclick - triggers on the double click of the mouse.
   - onmouseover - triggers when the mouse moves over an element.
   - onwheel - triggers when the wheel of the mouse moves over an element.
2. Keyboard events
   - onkeydown - triggers when a key is being pressed.
   - onkeypress - triggers when a key is pressed.
   - onkeyup - triggers when a key is released.
3. Window events
   - onload - triggers when a window is completely loaded.
   - onunload - triggers when a window is closed.
   - onresize - triggers when a window is resized.
4. Form events
   - onchange - triggers when the value of an element is changed.
   - onsumbit - triggers when a form is submitted.

- onreset - triggers when a form is reset.

5. Drag events

- ondrag - triggers when an element is dragged.
- ondrop - triggers when an element that is being dragged is dropped.

Summary

- HTML events make something happens. They are used just like other attributes.
- Different types of HTML events are mouse, keyboard, drag, window, media, and form events.
- The value of an event is written inside the quotes. What will happen next depends on the value.
- Generally, functions are triggered using HTML events because they can have multiple lines of codes inside them.

UNIT – III

Backend (Server-side) Development: Introduction to backend web development using backend web technologies and Frameworks

Backend development is the skill that powers the web. Backend development languages handle the 'behind-the-scenes' functionality of web applications. It's code that connects the web to a database, manages user connections, and powers the web application itself. Backend development works in tandem with the frontend to deliver the final product to the end user.

(a) What Is Back End Development?

As mentioned above, backend development is what keeps the internet running behind the scenes. Backend developers are primarily focused on how a website works. They write code that focuses on the functionality and logic powering the application they're working on, and the technology they work on is never directly seen by users. Backend technologies are a combination of Web servers, applications, and databases.

Responsibilities of backend programmers could involve writing APIs, writing code to interact with a database, creating libraries, working on business processes and data architecture, and much more

(b) Front-end vs Backend Programming

The key difference is that while backend developers build how a website functions, front-end programmers build and design the interface, determining how the site looks to users. Backend web development lays the foundational code that enables websites to process the actions that users take on the front end and deliver the correct information in return.

Neither works without the other—you need the back end to make the front-end work, and you need the front end so people can actually access and interact with the website.

There are also **full-stack** developers, who can work with both front-end and backend technologies. They're the jack-of-all-trades of the programming world.

(c) What Does a Backend Developer Do?

So, what does a backend developer actually do in a typical workday? As a back-end web developer, you might spend your day doing things like:

- Maintaining legacy applications (older applications that have already been built by other programmers)
- Putting out fires (e.g., bugs, broken applications)

- Attending meetings (to gather requirements for a project
- Collaborate with front end devs on a project, etc.)
- Actually writing code for new applications/projects

So, while you will certainly be coding with back-end languages, don't picture that as being the only thing you'll do as a back-end web developer.

(d) Types of Backend Development Languages

Backend programming can either be object-oriented (OOP) or functional. OOP is the technique that focuses on the creation of objects. With object-oriented programming, statements should be executed in a particular order. Popular OOP languages are Java, .NET, and Python.

Functional back-end programming is a technique that is more "action"-based. Functional programming uses declarative language, which means that statements can be executed in any order. It's commonly used for data science, and popular languages are SQL, F#, and R.

Back-end languages can either be statically typed or dynamically typed. The former is more rigid, but better at catching errors, whereas the latter is more flexible but allows for variables to change types (which could account for unexpected errors).

(e) Some Popular Backend Programming Languages

Now, let's take a look at some specific backend development languages and what they're used for.

BACKEND LANGUAGE	**KEY INFO**
1. Java	i) #3 most popular programming language in the world ii) Less beginner-friendly than other backend languages iii) Java developers make an average of $100,168/year
2. PHP	i) Powers 78.2% of all websites ii) Dynamically typed iii) Very forgiving of errors
3. .NET (C#, VB)	i) MVC (Model-View-Controller) architectural pattern ii) Can integrate with iOS, Linux, and Android iii) Highly stable and reliable
4. Ruby	i) Enables developers to create and launch apps quickly ii) Fantastic for prototyping iii) Grew in popularity in the early 2000s but has declined since then
5. Python	i) Fastest-growing programming language

	ii) Syntax is simple and easy to understand iii) Great for beginners
6. JavaScript	i) Can be used for both the front and back end ii) Very popular with a large community iii) Can be difficult to maintain and scale
7. TypeScript	i) Open source “superset” of JavaScript ii) 7th most used programming language iii) Developed by Microsoft to simplify JS code
8. Go	i) Statically typed, compiled programming language ii) Syntax is similar to C/C++ iii) Created by Google

(f) Other Backend Technologies to Know

Here are some other backend technologies that you may need to know that are not programming languages.

1. **Databases / and Cache**

   As a backend developer, it can be helpful to understand how databases and database caching work. Caching is a technique that stores frequently queried data in temporary memory. It supplements a primary database by removing unnecessary pressure on it.

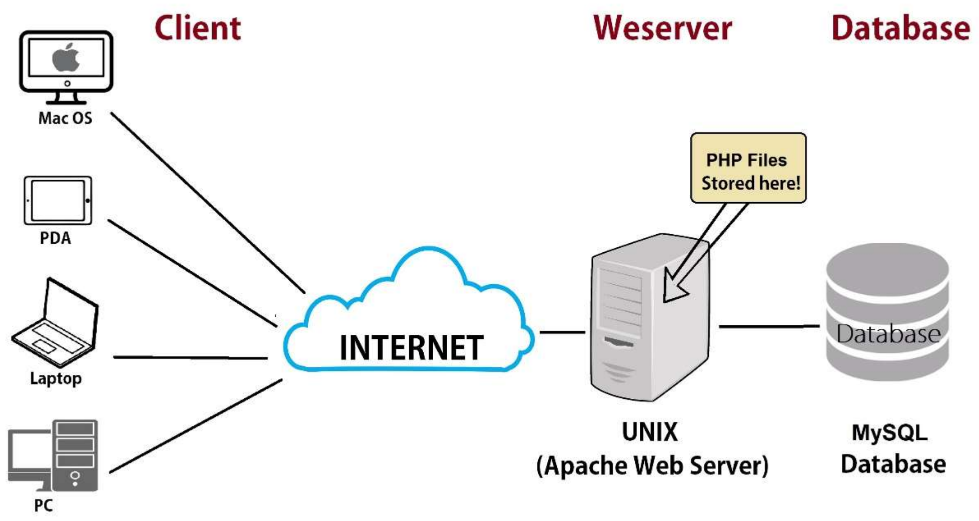
   Many backend job descriptions require knowledge of Database Management Systems (DBMS), such as MySQL, PostgreSQL, Microsoft Access, SQL Server, and caching mechanisms like Redis and Memcached.

2. **Web servers**

   A server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests. As a backend developer, you may need to know production web server technologies such as load balancers (e.g., Nginx and HTTP proxies).

   A **server** is a computer that provides data to other computers. The entire structure of the Internet is based upon a client- server model.

   Web server helps to deliver web content that can be accessed through the Internet. The most common use of web servers is to host websites, as the internet is not only used to fetch the information but there are other uses such as gaming, data storage or running business applications.



The primary job of a web server is to deliver web pages to clients. The communication between the client node and server node takes place using the Hypertext Transfer Protocol (HTTP). The delivered web pages include images, style sheets and scripts in addition to text content.

Server Software:

A web server commonly known as HTTP server or application server is a program that serves content using the HTTP protocol. This content is generally in the form of HTML documents, images, and other web resources. but can include any type of file. The content served by the web server can be pre- existing (static content) or generated on the fly (dynamic content).

i. **Apache web server- the HTTP web server:** Apache software foundation developed the free and open-source web server and can be installed and made to work on almost all operating systems including Linux, Unix, Windows, FreeBSD, Mac OS X and more. Almost, 60% of the web server machines run the Apache web server.
ii. **Apache Tomcat:** Apache Tomcat is free and open source What is a web server that can run on different operating systems like Linux, Unix, Windows, Mac OS X, Freebsd. It was developed to support servlets and JSP scripts. It can serve as a standalone server.

Services Provided by The Servers.

Today many services are being provided by the web server. Many of the server are based on cloud computing which is popular amongst the researchers, scientists & entrepreneurs.

Cloud Computing is distributing over a network, and has the ability to run a program or application on many connected computers at the same time.

It is used, where various computing concepts that involve a large number of computers are connected via real-time communication network such as the Internet. Various services provided by the web server are:

i. **Cost Efficient:** Web server is the most cost-efficient method to use, maintain and upgrade. Traditional desktop software costs companies a lot in terms of finance. On the other hand. It is available at much cheaper rates, Besides, there are many one-time-payment, pay-as-you-go and other salable options available, which makes it very reasonable for the company.
ii. **Resource Sharing:** Web Server has the capability to store unlimited information such as Google Drives, Cloud computing etc. The space where the data can be stored is shared by the other users at the same time like hard disk can be shared on physical network as LAN.
iii. **Data Sharing:** With the help of web servers one can easily access the information from anywhere, where there is an Internet connection using Google docs such as Documents, Excel sheets, Drawings, power point presentations etc.
iv. **Backup and Recovery:** As all the data nowadays is stored on web servers, backing it up and restoring the same is relatively much easier than storing the same on a physical device. Hence, the entire process of backup and recovery much simpler than other traditional methods of data storage.

Types of Servers

i. **Mail Server:** Mail Server provides a centrally-located pool of disk space for network users to store and share various documents in the form of emails. Since, all the data is stored in one location, administrators need only backup files from one computer.

ii. **Application Server:** An application server acts as a set of components accessible to the software developer through an API defined by the platform itself. For Web applications. These components are usually performed in the same running environment as its web server(s), and their main job is to support the construction of dynamic pages.

iii. **File Transfer Protocol (FTP) Server:** FTP uses separate control and data connections between the client and the server. FTP users may authenticate themselves in the form of a username and password. But can connect anonymously if the server is configured to allow it. For secure transmission username and password must be encrypted using FTP and SSL.

iv. **Database Server:** A database server is a computer program that provides database services to other computer programs or computers using client-server functionality, and some DBMSs (e.g., Mysql) depend on the client-server model for database access. Such a server is accessed either through a “front end” running on the user’s computer where the request is made or the “back end” where the request is served such as data analysis and storage.

v. **Domain Name System (DNS) Server:** A name server is a computer server that hosts a network service for providing responses to queries. It maps a numeric identification or addressing component. This service is performed by the server in response to a network service protocol request.

The primary function of these DNS servers is the translation (resolution) of human-memorable domain names and host names into the corresponding numeric Internet protocol (IP) addresses. The secondary function of DNS is to recognize a namespace of the Internet, used to identify and locate computer systems and resources on the Internet.

3. API (REST & SOAP)

APIs, Application programming interfaces, allow two separate applications to talk to each other. They play a role in how most server-side software architectures are built. Two API styles you might come across as a backend developer are SOAP and REST. SOAP stands for Simple Object Access Protocol. REST stands for Representational State Transfer.

(g) PHP

PHP powers 78.2% of all websites whose server-side programming language we know. The language was first released in 1995 when there were few options for building dynamic websites.

Since PHP is dynamically typed, it means you're able to come up with a variety of solutions and workarounds for one problem. It also means that the same bit of code can mean something different depending on the context, which makes programs written in PHP tricky to scale and sometimes slow to run.

PHP is a great language to learn for those who are just starting out for a number of reasons:

i. It's more forgiving of errors, which means that you're able to compile and run a program until you reach a problematic part.
ii. There is an abundance of resources dedicated to the language as a result of the large community and tool support. The language undergoes updates, so ensure you're learning from an up-to-date tutorial.
iii. The setup is relatively easy compared to a language like Ruby on Rails. You can download XAMPP, MAMP (for Macs) or WAMP (for Windows) and you should be ready to go in 5 minutes.

What You Can Do with PHP

According to the PHP website, you can:

i. Collect form data
ii. Generate dynamic page content
iii. Send and receive cookies
iv. Write command line scripting
v. Write server-side scripting
vi. Write desktop applications

Companies That Use PHP

Here are some companies using PHP in their dev teams:

i. Facebook
ii. Lyft
iii. Mint
iv. Hootsuite
v. Viber
vi. Buffer
vii. DocuSign

PHP was designed by Rasmus Lerdorf to display his resume online and to collect data from his visitors.

Basically, PHP allows a static web document to become dynamic. "PHP" is a recursive acronym that stands for "PHP: Hypertext Preprocessor". PHP preprocesses (that is, PHP processes before the output is sent to the browser) hypertext documents. Because of this, the pages can change before the user sees them, based on conditions. This can be used to write something to the page, create a table with a number of rows equal to the number of times the user has visited, or integrate the web page with a web database, such as MySQL.

(h) Hello World

"Hello world." is the first program most beginning programmers will learn to write in any given language. Here is an example of how to print "Hello world!" in PHP.

Code:

```
<?php
  echo "Hello world!";
?>
```

Output:

```
Hello world!
```

This is as basic as PHP gets. Three simple lines, the first line identifies that everything beyond the <?php tag, until the ?> tag, is PHP code. The second line causes the greeting to be printed (or echoed) to the web page. This next example is slightly more complex and uses variables.

Hello World With Variables

This example stores the string "Hello world!" in a variable called $string. The following lines show various ways to display the variable $string to the screen.

Code:

```
<?php
  // Declare the variable 'string' and assign it a value.
  // The <br /> is the HTML equivalent to a new line.
  $string = 'Hello world!<br />';
  // You can echo the variable, similar to the way you would
echo a string.   echo $string;


  // You could also use
print.   print $string;


  // Or, if you are familiar with C, printf can be used too.

  printf('%s', $string);

?>
```

Output:

```
Hello world!<br />Hello world!<br />Hello world!<br />
```

HTML Render:

Hello world!

Hello world!

Hello world!

The previous example contained two outputs. PHP can output HTML that your browser will format and display. The PHP Output box is the exact PHP output. The HTML Render box is approximately how your browser would display that output. Don't let this confuse you, this is just to let you know that PHP can output HTML. We will cover this much more in depth later.

(i) Variables

Variables are the basis of any programming language: they are "containers" (spaces in memory) which hold data. The data can change, thus it is "variable".

If you've had any experience with other programming languages, you know that in some of the languages, you must define the type of data that the variable will hold. Those languages are called statically-typed, because the types of variables must be known before you store something in them. Programming languages such as C++ and Java are statically-typed. PHP, on the other hand, is dynamically-typed, because the type of the variable is linked to the value of the variable. You could define a variable for a string, store a string, and then replace the string with a number. To do the same thing in C++, you would have to cast, or change the type of, the variable, and store it in a different "container".

All variables in PHP follow the format of a dollar sign ($) followed by an identifier i.e. $variable_name. These identifiers are case-sensitive, meaning that capitalization matters, so $wiki is different from $Wiki.

(j) Real world analogy

To compare a variable to real world objects, imagine your computer's memory as a storage shed. A variable would be a box in that storage shed and the contents of the box (such as a cup) would be the data in that variable.

If the box was labeled kitchen stuff and the box's contents were a cup, the PHP code would be:

$kitchen_stuff = 'cup';

If I then went into the storage shed, opened the box labeled kitchen stuff, and then replaced the cup with a fork, the new code would be: *$kitchen_stuff = 'fork';*

Notice the addition of the = in the middle and the ; at the end of the code block. The = is the assignment operator, or in our analogy, instructions that came with the box that states "put the cup in the box". The ; indicates to stop evaluating the block of code, or in our analogy, finish up with what you are doing and move on to something else.

Also notice the cup was wrapped in single quotes instead of double. Using double quotes would tell the PHP parser that there may be more than just a cup going into the box and to look for additional instructions.

```
$bathroom_stuff = 'toothbrush';

$kitchen_stuff = "cup $bathroom_stuff";

//$kitchen_stuff contents is now '''cup toothbrush'''
```

Single quotes tell the PHP parser that it's only a cup and to not look for anything more. In this example the bathroom box that should've had its contents added to the kitchen box has its name added instead. $bathroom_stuff = 'toothbrush';

$kitchen_stuff = 'cup $bathroom_stuff';

//$kitchen_stuff contents is now '''cup $bathroom_stuff'''

So again, try to visualize and associate the analogy to grasp the concept of variables with the comparison below. Note that this is a real world object comparison and NOT PHP code.

Computer memory (RAM) = storage shed

Variable = a box to hold stuff

Variable name = a label on the box such as kitchen stuff

Variable data = the contents of the box such as a cup

Notice that you wouldn't name the variable box, as the relationship between the variable and the box is represented by the code>$ and how the data is stored in memory. For example a constant and array can be considered a type of variable when using the box analogy as they all are containers to hold some sort of contents, however, the difference is on how they are defined to handle the contents in the box.

Variable: a box that can be opened while in the storage shed to exchange the contents in the box.

Constant: a box that cannot be opened to exchange its contents. Its contents can only be viewed and not exchanged while inside the storage shed.

Array: a box that contains 1 or more additional boxes in the main box. To complicate matters for beginners, each additional box may contain a box as well. In the kitchen stuff box we have two boxes, the clean cup box $kitchen_stuff["clean_cup"] = 'the clean cup';

and the dirty cup box

$kitchen_stuff["dirty_cup"] = 'the dirty cup';

More on variables, from the PHP manual

The print and echo statements

Print is the key to output. It sends whatever is in the quotes (or parentheses) which follow it to the output device (browser window). A similar function is echo, but print allows the user to check whether or not the print succeeded.

When used with quotation marks, as in:

print "Hello, World!";

The quoted text is treated as if it were a string, and thus can be used in conjunction with the concatenation (joining two strings together) operator as well as any function that returns a string value.

The following two examples have the same output.

print "Hello, World!";

and

print "Hello" . ", " . "World!";

The dot symbol concatenates two strings. In other programming languages, concatenating a string is done with the plus symbol and the dot symbol is generally used to call functions from classes.

Also, it might be useful to note that under most conditions echo can be used interchangably with print. print returns a value, so it can be used to test if the print succeeded, while echo assumes everything worked. Under most conditions there is nothing we can do if echo fails.

The following examples have the same output again.

echo "Hello, World!";

and

echo "Hello" . ", " . "World!";

We will use echo in most sections of this book, since it is the more commonly used statement.

It should be noted that while echo and print can be called in the same way as functions, they are, in fact, language constructs, and can be called without the brackets. Normal functions (almost all others) must be called with brackets following the function identifier.BASICS

The Examples

Example 1 - Basic arithmetic operators

This example makes use of the five basic operators used in mathematical expressions. These are the foundation of all mathematical and string operations performed in PHP.

The five mathematical operators all function identically to those found in C++ and Java add (+) subtract (-) multiply (*) divide (/) assign (=)

Examine this example. Each mathematical expression to the right of the assign operator is evaluated, using the normal order of operations. When the expression has been evaluated, the resultant value is assigned to the variable named to the left of the assign operator.

PHP Code:

```
<?php

$x = 25;

$y = 10;

$z = $x + $y;

echo $z;

echo "<br />";

$z = $x / $y;
```

```
echo $z;

echo "<br />";

$z = $y * $y * $x;

echo $z - 1250;

echo "<br />";

?>
```

PHP Output:

```
35<br />2.5<br />1250<br />
```

HTML Render:

```
35

2.5

1250
```

Note: If you are not familiar with HTML, you may not know the purpose of this part of the above code: echo "<br />";

Its purpose is to insert an HTML "line break" between the results, causing the browser to display each result on a new line when rendering the page. In the absence of this line, the above code would instead print: 352.51250

This is of course not the desired result.

There are two code options which perform the opposite of the assign (=) operator. The keyword null should be used for variable nullification, which is actually used with the assign (=) operator in place of a value. If you want to destroy a variable, the unset() language construct is available.

Examples:

$variable = null;

or unset($variable);

Example 2 - String concatenation

This example demonstrates the concatenation operator (.), which joins together two strings, producing one string consisting of both parts. It is analogous to the plus (+) operator commonly found in C++ string class (see STL), Python, Java, JavaScript implementations.

The statement

```
$string = $string . " " . "All the cool kids are doing it.";
```

prepends the current value of $string (which is "PHP is wonderful and great.") to the literal string " All the cool kids are doing it." and assigns this new string to $string.

Code:

```
<?php

  $string = "PHP is wonderful and great.";

  $string = $string . " " . "All the cool kids are doing it.";    echo
$string;

?>
```

Output:

```
PHP is wonderful and great. All the cool kids are doing it.
```

Example 3 - Shortcut operators

This snippet demonstrates self-referential shortcut operators. The first such operator is the ++ operator, which increments $x (using the postfix form) by 1 giving it the value 2. After incrementing $x, $y is defined and assigned the value 5.

The second shortcut operator is *=, which takes $y and assigns it the value $y * $x, or 10.

After initializing $z to 180, the subsequent line performs two shortcut operations. Going by order of operations (see manual page below), $y is decremented (using the prefix form) and divided into $z. $z is assigned to the resulting value, 20.

Code:

```
<?php

$x = 1;    $x++;
echo  $x  .  "  ";
$y = 5;    $y *=
$x;    echo $y .
" ";   $z = 180;
$z /= --$y;

  echo $z; ?>
```

Output:

```
2 10 20
```

Note: The expanded version of the above code (without the shortcut operators) looks like this:

```
<?php

  $x = 1;    $x =
$x + 1;    echo
$x . " ";    $y =
5;    $y = $y *
$x;  echo $y . "
";   $z = 180;

  $y = $y - 1;   $z
= $z / $y;

  echo $z; ?>
```

The output is the same as seen in the above example.

(k) Operators

An operator is any symbol used in an expression used to manipulate data. The seven basic PHP operators are:

= (assignment)

+ (addition)

- (subtraction)

* (multiplication)

/ (division)

% (modulus)

. (concatenation)

In addition, each of the above operators can be combined with an assignment operation, creating the operators below: += (addition assignment)

-= (subtraction assignment)

*= (multiplication assignment)

/= (division assignment)

%= (modulus assignment)

.= (concatenation assignment)

These operators are used when a variable is added, subtracted, multiplied or divided by a second value and subsequently assigned to itself.

In other words, the statements

$var = $var + 5;

and

`$var += 5;` are
equivalent.

There are also increment and decrement operators in PHP.

++ (increment) --
(decrement)

These are a special case of the addition and subtraction assignment operators.

This code uses the addition assignment operator to increment and decrement a variable.

Code:

```
$var = 0;

$var += 1; echo "The incremented
value is $var.\n";

$var -= 1; echo "The decremented
value is $var.\n";
```

Output:

The incremented value is 1.

The decremented value is 0.

While this is perfectly legal in PHP, it is somewhat lengthy for an operation as common as this. It can easily be replaced by the increment operator, shortening the statement. This code snippet uses the increment and decrement operators to increase and decrease a variable's value by one.

Code:

```
$var = 3;

$var++;  echo  "The  incremented
value is $var.\n";
```

```
$var--;  echo  "The  decremented
value is $var.\n";
```

Output:

The incremented value is 4.

The decremented value is 3.

Using the increment operator makes your code slightly easier to read and understand.

For a more in-depth overview of PHP's operators, including an explanation of bitwise operators, refer to the manual link below.

Newline and Other Special Characters

Both of the below examples make use of the newline character (\n) to signify the end of the current line and the beginning of a new one.

The newline is used as follows:

Code:

echo "PHP is cool,\nawesome,\nand great.";

Output:

PHP is cool,

awesome,

and great.

Notice: the line break occurs in the output wherever the \n occurs in the string in the echo statement. However, a \n does not produce a newline when the HTML document is displayed in a web browser. This is because the PHP engine does not render the script. Instead, the PHP engine outputs HTML code, which is subsequently rendered by the web browser. The linebreak \n in the PHP script becomes HTML whitespace, which is skipped when the web browser renders it (much like the whitespace in a PHP script is skipped when the PHP engine generates HTML). This does not mean that the \n operator is useless; it can be used to add whitespace to your HTML, so if someone views the HTML generated by your PHP script they'll have an easier time reading it.

In order to insert a line-break that will be rendered by a web browser, you must instead use the <br /> tag to break a line.

Therefore the statement above would be altered like so: echo 'PHP
is cool,<br />awesome<br />and great.';

The function nl2br() is available to automatically convert newlines in a string to <br /> tags.

The string must be passed through the function, and then reassigned:

PHP Code:

```
$string = "This\ntext\nbreaks\nlines.";

$string = nl2br($string); print
$string;
```

PHP Output:

```
This<br /> text<br /> breaks<br /> lines.
```

HTML Render:

This

text

breaks

lines.

Additionally, the PHP output (HTML source code) generated by the above example includes linebreaks.

Other special characters include the ASCII NUL (\0) - used for padding binary files, tab (\t) - used to display a standard tab, and return (\r) - signifying a carriage return. Again, these characters do not change the rendering of your HTML since they add whitespace to the HTML source. In order to have tabs and carriage returns rendered in the final web page, &tab; should be used for tabs and <br /> should be used for a carriage return.

(l) Webservers

On webservers, information sent to a PHP app may either be a GET operation or a POST operation.

For a GET operation, the parameters are sent through the address bar. Parameters within the bar may be retrieves by using accessing $_GET['parameter']. On a POST operation, submitted input is accessed by $_POST['parameter'].

A more generic array, $_REQUEST['parameter'] contains the contents of both $_GET, $_POST, and $_COOKIE.

(m) Comments

As you write more complex scripts, you'll see that you must make it clear to yourself and to others exactly what you're doing and why you're doing it. Comments and "good" naming can help you make clear and understandable scripts because:

When writing a script takes longer than a week, by the time you're done, you won't remember what you did when you started, and you will most likely need to know. Any script that is commonly used will need rewriting sooner or later. Rewriting is much easier (and in many cases, made possible) when you write down what you did. If you want to show someone your scripts, they should be nice and neat.

(n) Comments

Comments are pieces of code that the PHP parser skips. When the parser spots a comment, it simply keeps going until the end of the comment without doing anything. PHP offers both one line and multi-line comments.

(o) One-Line Comments

One-line comments are comments that start where ever you start them and end at the end of the line. With PHP, you can use both // and # for your one-line comments (# is not commonly used). Those are used mainly to tell the reader what you're doing the next few lines. For example:

```
 //Print the variable $message   echo
$message;
```

It's important to understand that a one-line comment doesn't have to 'black out' the whole line, it starts where ever you start it. So it can also be used to tell the reader what a certain variable does:

```
 $message = ""; //This sets the variable $message to an empty string
```

The $message = ""; is executed, but the rest of the line is not.

(p) Multi-Line Comments

This kind of comment can go over as many lines as you'd like, and can be used to state what a function or a class does, or just to contain comments too big for one line. To mark the beginning of a multiline comment, use /* and to end it, use */ . For example:

```
/* This is a     multiline
comment

   And it will close

   When  I  tell  it  to.
*/
```

You can also use this style of comment to skip over part of a line. For example:

```
$message = ""/*this would not be executed*/;
```

Although it is recommended that one does not use this coding style, as it can be confusing in some editors.

(q) Naming

Naming your variables, functions and classes correctly is very important. If you define the following variables:

```
$var1 = "PHP";  $var2
= 15;
```

They won't say much to anyone. But if you do it like this:

```
$programming_language = "PHP";

$menu_items = 15;
```

It would be much clearer. But don't go too far. $programming_language, for example is not a good name. It's too long, and will take you a lot of time to type and can affect clarity. A better name may be $prog_language, because it's shorter but still understandable. Don't forget to use comments as well, to mark what every variable does.

```
$prog_language = "PHP";  //The programming language used to write this script

$menu_items = 15;        //The maximum number of items allowed in your personal menu
```